UNIVERSIDAD DEL AZUAY

Facultad de Ciencias de la Administración

Escuela de Ingeniería de Sistemas

Cluster de Alta Disponibilidad para Servidores Web en LINUX

Monografía previa a la obtención del Título de Ingeniero de Sistemas.

Autores: **Stanley Cortéz Guerrero**

Mónica Galarza Rodas

Directores: **Ing. Fernando Balarezo R.**

Ing. Pablo Esquivel L.

Cuenca - Ecuador

2005

AGRADECIMIENTOS

Agradecemos a la Universidad del Azuay y a nuestros profesores, que con su paciencia, esfuerzo y grandes conocimientos brindados, supieron encaminarnos en el logro de nuestros objetivos.

DEDICATORIA

A mis queridos padres, porque han estado junto a mí a lo largo de todo este trayecto, apoyándome con amor y dedicación, para la plena realización de todos mis anhelos.

Mónica

DEDICATORIA

Esta monografía es dedicada a mi Madre, que con su abnegado esfuerzo supo guiarme, apoyarme y ayudarme en el cumplimiento de uno de mis grandes sueños.

Stanley

Ideas y opiniones vertidas en la presente monografía son de exclusiva responsabilidad de sus autores.

Mónica Galarza Rodas *Stanley Cortéz Guerrero*

INTRODUCCION

Tanto las aplicaciones Web como servicios de mail de una organización suelen realizar tareas fundamentales, en las que el costo de su falta de disponibilidad puede superar rápidamente el costo de la aplicación y la infraestructura. Es importante que estas aplicaciones proporcionen una alta disponibilidad y confiabilidad. La manera más sencilla y efectiva de conseguir disponibilidad y confiabilidad consiste en replicar zonas críticas en el sistema.

Si un sitio tiene más de un servidor y se produce un fallo en alguno de ellos, es posible redirigir las solicitudes de procesamiento a otro servidor. Con esto se consigue un sitio Web de alta disponibilidad.

Al incrementar el número de servidores Web y Mail, el sitio podrá también admitir cargas mucho mayores sin mayores problemas, es decir al agregar servidores se puede ofrecer una escalabilidad casi lineal (lo que significa que el rendimiento aumenta) cuando se siguen las directrices para un buen diseño de clúster.

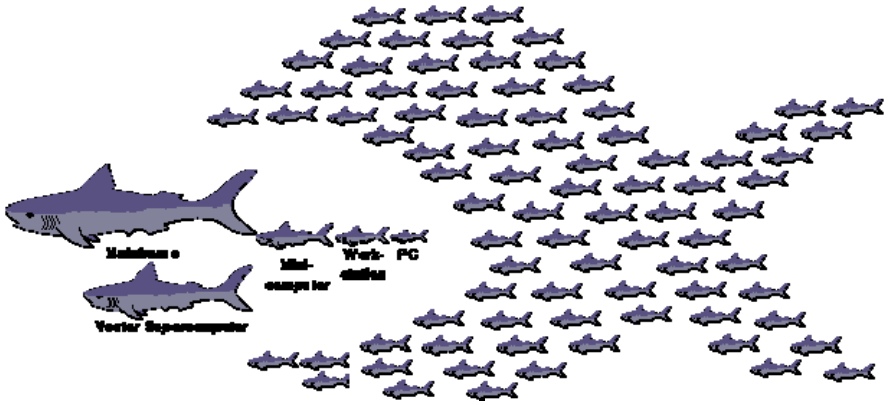
Esta monografía tratará sobre los principios básicos de clustering, enfocándonos principalmente en los Clusters de Alta Disponibilidad (High Avality ó HA), y así, tener un conocimiento general de las ventajas y desventajas que representan, cada uno de sus componentes, técnicas, topologías y herramientas en la implementación de dicho cluster.

Además de enumerar las características que hacen a Linux un Sistema Operativo eficiente y robusto, para implementar un Cluster HA de Servidores Web y Mail.

Para reforzar todos estos conocimientos, se indican todos los pasos que se deben seguir para la implementación, de un caso sencillo de Cluster de Alta Disponibilidad para Servidores Web y Mail.

CAPITULO I: CLUSTER

1.1 ¿Qué es Cluster?



Un Cluster es una clase de arquitectura de computador paralelo, que se basa en unir equipos independientes integrados por medio de redes de interconexión, los cuales ejecutan una serie de aplicaciones de forma conjunta, que aparecen ante clientes y aplicaciones como un solo sistema, es decir se utiliza un cluster con varios computadores para crear un supercomputador.

Los clusters han evolucionado para apoyar actividades en aplicaciones que van desde supercómputo y software de misiones críticas, servidores Web y comercio electrónico, bases de datos de alto rendimiento.

1.2 ¿Qué es un Nodo?

Un nodo es cada uno de los ordenadores que forma parte de un cluster. Se excluye de esta definición a los hubs, switchs, routers y cualquier otro dispositivo

de interconexión que no participe en la ejecución de los procesos ni puedan migrarse procesos hacia él.

1.3 Características de los Clusters.

En su parte central, la tecnología de Clusters consta de 3 partes:

1. Un cluster debe estar compuesto de 2 o más nodos, capaces de ejecutar las aplicaciones ofertadas por el mismo. Un cluster no requiere necesariamente ser homogéneo, es decir, no importa que los nodos sean diferentes, ya que la abstracción a la que se ven sometidos los habilita para formar parte de un mismo cluster.

2. El segundo componente, hace referencia al sistema operativo confeccionado especialmente para esta tarea, un conjunto de compiladores y aplicaciones especiales, que permiten que los programas que se ejecutan sobre esta plataforma tomen las ventajas de esta tecnología de Clusters.

3. Y el tercer componente es la interconexión de hardware entre las máquinas (nodos) del Cluster. Se han desarrollado interfaces de interconexión especiales muy eficientes, pero comúnmente las interconexiones se realizan mediante una red Ethernet dedicada de alta velocidad. Es mediante esta interfaz que los nodos del Cluster intercambian entre si asignación de tareas, actualizaciones de estado y datos del programa. Existe otra interfaz de red que conecta al Cluster con el mundo exterior.

Esto excluye a los SMP, los Mainframes y a un sólo nodo, ya que no disponen de red y no se ajustan a la definición de cluster.
Una máquina SMP (Symetric MultiProcessing) sigue una modelo de paralelismo al utilizar más de un microprocesador simultáneamente. La memoria y el disco son igualmente accesibles desde cualquiera de los procesadores.

Un Mainframe es supercomputador con varios procesadores trabajando en paralelo, cuya memoria puede ser compartida por todos los procesadores o poseer cada cual la suya propia. La interconexión se realiza mediante un bus de datos cuya velocidad supera con creces la de cualquier red.

1.4 Ventajas y Desventajas de los Clusters

1.4.1 Ventajas

- Relación coste/prestaciones.
- Flexibilidad.
- Disponibilidad.
- Escalabilidad
- Incorporación de tecnología Punta.
- Heterogeneidad.

1.4.2 Desventajas

- Software.
- Problemas de administración y gestión.
- Memoria físicamente distribuida => utilización menos eficiente.
- Varias copias del sistema operativo.
- La red es el cuello de botella del sistema.

1.5 Clusters vs Supercomputadores

Las principales ventajas entre un cluster y un superordenador como puede ser un Mainframe son: precio, reutilización, escalabilidad.

En primer lugar, el precio de un cluster con una capacidad de cálculo similar a la de un Mainframe es un 10% menor, lo que permite el ahorro de esa inversión para la adquisición de nodos más potentes.

La posibilidad de reutilizar máquinas antiguas para la construcción del cluster y su gran escalabilidad son circunstancias que favorecen la decisión del sector

empresarial en la adquisición de un cluster, ya que cuando un Mainframe se queda "pequeño" respecto a su capacidad de cálculo no hay otra solución que comprar un nuevo Mainframe. [1]

1.6 Escalabilidad

La escalabilidad permite acomodar los recursos y el rendimiento de acuerdo a las necesidades que existan. La escalabilidad crece hacia arriba, cuando los sistemas aumentan en tamaño, costos, rendimiento y recursos. La escalabilidad hacia abajo, hace referencia a reducir costos.

La escalabilidad ideal es de complejidad lineal, la cual aumenta en N el número de elementos en el sistema, del cual se espera un rendimiento N veces al inicial y un coste N veces el coste inicial. Esto se aproxima a la realidad en los sistemas distribuidos ya que el precio de obtener el doble de procesamiento en uno de estos sistemas es ligeramente superior al comprar una máquina el doble.[1]

1.7 Tipos de Cluster.

Existen 3 tipos de clusters de acuerdo a su utilidad: de alto rendimiento, de alta disponibilidad y de balanceo de carga.

1.7.1 Clusters de Alto Rendimiento

Los clusters de alto rendimiento se basan en un conjunto de máquinas que logran una capacidad de cálculo máxima al repartirse la carga de los procesos entre los nodos de acuerdo a determinadas reglas. Con esto se consigue mejorar el rendimiento en la obtención de la solución de un problema, es decir este tipo de clusters se aplica mejor en problemas grandes y complejos que requieren una cantidad enorme de potencia computacional.

Los procesos que se ejecutan en un cluster corren bajo un entorno de procesamiento paralelo. El procesamiento paralelo consiste en dividir grandes tareas en pequeñas subtareas que son procesadas paralelamente.

Entre las aplicaciones más comunes de clusters de alto rendimiento se encuentran el pronóstico numérico del estado del tiempo, astronomía, investigación en criptografía, análisis de imágenes, y más.
Estos procesos consumen muchos recursos de cómputo y es necesario utilizar supercomputadoras que puedan satisfacer las necesidades de los investigadores, por esta razón los clusters de alto rendimiento con software libre y con la reutilización de equipos con pocas características, reducen en gran medida la inversión que se tendrá que realizar con relación a una supercomputadora, además de presentar otras ventajas como la escalabilidad y flexibilidad, aunque también tienen desventajas como la utilización un mayor espacio físico y requerimientos de infraestructura humana mejor capacitada.

El uso del cómputo de alto rendimiento en el área científica se ha propagado de manera tal que se ha generado una rama dedicada al cómputo científico y con esto han surgido nuevas metodologías.

El ejemplo más claro de este tipo, es el proyecto Beowulf, construido en la NASA en 1994, fue el primer cluster de la historia, y su finalidad era el cálculo masivo de datos.

Otro ejemplo es MOSIX, que consiste en unos parches para el núcleo de Linux, con lo que se logra utilizar de forma transparente toda una red de equipos como si fuera una única supercomputadora, permitiendo la migración transparente de procesos y la compartición de recursos entre máquinas.

1.7.2 Cluster de Alta Disponibilidad

En este tipo de cluster no busca exactamente conseguir una gran potencia de cálculo, si no mas bien conseguir un conjunto de máquinas que todas realicen la misma función y que, ante el fallo de una de ellas, las demás puedan asumir sus tareas de una forma transparente y rápida.
Por supuesto, la escalabilidad también es importante, ya que siempre podremos añadir más máquinas al cluster para así conseguir más potencia, pero el objetivo primordial no es este, sino la resistencia a cualquier fallo imprevisto.

1.7.3 Clusters de Balanceo de Carga

En este tipo de Cluster, existe una jerarquía de nodos maestros y nodos esclavos. Los maestros reciben la petición del servicio y mandan la tarea a los esclavos para que la ejecuten. Estos devuelven el resultado y el servidor responde. Todos los clusters mencionados anteriormente hacen uso de un sistema de balanceo de carga que reparte los procesos entre todos los nodos del cluster de forma que no haya ningún nodo saturado.

1.8 Principios en los que se basan los clusters

1.8.1 Procesamiento paralelo

Por procesamiento paralelo se entiende la capacidad de utilizar varios procesadores para ejecutar diferentes partes del mismo programa simultáneamente. El objetivo principal del paralelismo es reducir el número de ciclos de ejecución de un programa con relación al número de procesadores que existen en el sistema, es decir, "divide y vencerás". En esta idea es en la que se apoyan los Clusters de alto rendimiento para lograr su cometido.

1.8.2 Sistema Distribuido

Un sistema distribuido es un conjunto de ordenadores o procesadores independientes (nodos) que de cara al usuario funcionan como uno solo. Está formado por varios componentes que cooperan estrechamente para dar un servicio único.

Características de los Sistemas Distribuidos.

- Cada elemento de cómputo tiene su propia memoria y su propio Sistema Operativo.
- Control de recursos locales y remotos.

- Sistemas Abiertos (Facilidades de cambio y crecimiento).
- Plataforma no standard (Unix, NT, Intel, RISC, Etc.).
- Medios de comunicación (Redes, Protocolos, Dispositivos, Etc.).
- Capacidad de Procesamiento en paralelo.

1.9 ¿Por qué elegir Linux para la implementación de un Cluster?

La elección de Linux como sistema operativo para un sistema de clustering se ha basado en las siguientes características:

La evolución y estabilidad que ha alcanzado el SO Linux, ha contribuido importantemente al desarrollo de muchas tecnologías nuevas, entre ellas la de Clusters.

Linux es conocida, como una plataforma de computación estable y supercomputación muy consolidada. Se puede afrontar la idea de eliminar SPF (Punto Simple de Fallo) en los sistemas de computación, pero sólo a partir de un nivel que Linux como sistema operativo pueda obrar.

Linux posee una de las pilas TCP/IP más completas y estables que existen actualmente. La característica de IP Aliasing de Linux, permite asignar varias direcciones IP a una misma interfaz; esto nos permite poder levantar una dirección IP, en un nodo del Cluster. Con estos servicios el Cluster puede ser consciente de la topología de la red y reaccionar ante las caídas de líneas de comunicaciones o nodos.

Actualmente el kernel de Linux soporta RAID 0,1 y 5 con el driver MD. Además de estas ventajas que nos da Linux, se añaden otras muy importantes, como que Linux, es un software libre, lo que representa menor costo en la implementación.

Adicionalmente, la naturaleza de fuente abierta del sistema Linux ha permitido a los programadores, añadir directamente características adicionales al sistema operativo para responder a las necesidades en los Clusters de computación. Dándonos por entender que Linux es un sistema muy evolutivo y ajustable.

Los sistemas UNIX propietarios como Solaris, HPUX,… requieren hardware propietario y por ende nos lleva a un elevado costo.

Linux se preocupa para no dejar obsoleto el hardware que no es de última tecnología.

También cabe destacar el enorme esfuerzo que está realizando la comunidad Linux mundial en la difusión de documentación, ya sea a nivel de HOWTO's, F.A.Q (Frecuently Asked Questions), manuales, tutoriales paso a paso para principiantes o textos especializados para expertos en la materia.

1.10 ¿Quiénes utilizan clusters?

Entre algunas de las instituciones que utilizan clusters podemos citar:

- Google. Es una compañía líder en indexar y desarrollo de nuevas tecnologías de la información del Internet. Algunas características importantes de la tecnología utilizada:
  - Atiende en promedio más de 200 millones de consultas por día (2315 consultas por segundo).
  - Almacena la información indexada de más de 3 mil millones de páginas Web en 36 idiomas y más de 35 millones de documentos en formatos distintos a HTML accesibles a través de su máquina de búsqueda.
  - Opera un Cluster de más de 10.000 sistemas Linux.
  - Manejo de una base de datos muy eficiente con cerca de 700 millones de noticias del Usenet (20 años de información y 1 TB de datos) [2]

- WebArchive.Org. Es una Hemeroteca digital del Internet, en cuyos acervos se encuentra la información de más de 5 años de las

páginas Web, representando una base de datos con más de 100TB de información. Es a través de su Máquina de Tiempo (The Wayback Machine) que se pueden consultar más de 10 mil millones de páginas. La tecnología de cluster utilizada por la Máquina del Tiempo consta de un Cluster de cerca de 400 máquinas. [2]

- NASA
- NOAA (The Nacional Oceanic and Atmospheric Administration)
- Yahoo.com
- Instituto Nacional de Salud Unidos (Beowulf 675 nodos, 1350 procesadores).
- Universidades en todo el mundo

CAPITULO II: CLUSTER DE ALTA DISPONIBILIDAD

2.1 ¿Qué es Alta Disponibilidad?

Ya que en la actualidad estamos cada vez mas demandados por la creciente globalización del mundo empresarial, a su vez la persistente necesidad de mantener la competitividad de las empresas, el movimiento de datos de todo tipo en Internet (más de un 100% anual) y la incuestionable importancia de la informática en las empresas actuales de cualquier tamaño, es cada día más importante que los sistemas informáticos de éstas puedan funcionar de forma ininterrumpida y sin errores las 24h del día, 7 días a la semana y 365 días al año, A esta necesidad de un servicio ininterrumpido y fiable se le conoce como alta disponibilidad. [3]

Algunas cosas que precisan alta disponibilidad son aquellos servicios que hacen que una determinada empresa pueda funcionar:

- El Sitio de Intranet
- Los servicios de Correo
- El Servicio DNS

Estos servicios pueden fallar por dos motivos

- Mal comportamiento de software
- Mal comportamiento de hardware

Aunque Linux es un Sistema Operativo bastante confiable, no se puede confiar tanto en las empresas que fabrican hardware, ya que si algún dispositivo falla, el sistema no será usable y por tanto no cumplirá con sus objetivos. Para prevenir estos fallos en hardware, se debe tomar muchas precauciones al momento de comprarlo.

A los usuarios, no les interesa cual fue el motivo del fallo, lo único que a ellos les preocupa es que el **"SERVICIO"** debe estar **"DISPONIBLE".**

La disponibilidad de un sistema depende de varios factores, tales como, el tiempo que el sistema está funcionando sin problemas, el tiempo en el que el sistema esta fallando, y por último el tiempo que se tarda en reparar o restaurar el sistema. Para medir todos estos factores, se necesitan fallos. Existen dos tipos de fallos: los fallos que provocan los administradores para ver o medir los tiempos de recuperación y tiempos de caídas, y los fallos no provocados que son los que demuestran que los tiempos de reparación suelen ser mucho más grandes de los que se estimó en los fallos provocados.

2.2 Sistemas de Alta Disponibilidad y Sistemas Tolerantes a Fallos

En un sistema tolerante a fallos, cuando se produce un fallo en hardware, el hardware asociado a este tipo de sistema es capaz de detectar el subsistema que falla y obrar en consecuencia para restablecer el servicio en segundos (o incluso décimas de segundo). El cliente del servicio no notará ningún tiempo de fuera de servicio.

En los sistemas de alta disponibilidad existen los tiempos de fuera de servicio; son mínimos pero existen, van desde 1 minuto hasta 5 o 10 minutos, según sea el caso. En teoría esta es la única diferencia entre ambos, pero en los últimos años, se ha ido acercando la idea de alta disponibilidad a la idea de tolerancia a fallos, debido al abaratamiento de hardware, y de ciertas tecnologías que han ido surgiendo.

En la mayoría de los análisis, que se hacen de un sistema de servicio, si la aplicación puede estar un mínimo tiempo fuera de servicio, y podemos permitir que el cliente pierda la sesión o la conexión, temporalmente, la alta disponibilidad es una opción muy apropiada. Hay soluciones de alta disponibilidad en las cuales las conexiones se mantienen y las sesiones se recuperan.

2.3 SPF (Single Point Failure ó Punto Simple de Fallo)

La regla básica para implementar un sistema de alta disponibilidad es la replicación de elementos. Con SPF se quiere hacer referencia a cualquier elemento no replicado y que puede estar sujeto a fallos, afectando con ello al servicio.

Si en un entorno de servidores falla uno y no puede ser reemplazado fácilmente por otro, en los tiempos anteriormente mencionados, el servidor se considerará como SPF.

En el caso de alta disponibilidad (con hardware y software adecuado) se puede reemplazar un adaptador de red o un servidor automáticamente. No sólo los servidores han de ser redundantes si no los elementos que facilitan el servicio, como routers, bridges o la propia red local.

Es conveniente olvidar la redundancia a cierto nivel, pues habrá un nivel en que la alta disponibilidad se hace extremadamente cara.

2.4 Gestión de Almacenamiento

Una de las primeras cosas en las que se tendrá que pensar al momento de implantar un sistema de alta disponibilidad, es cómo asegurar la integridad y fiabilidad de los datos almacenados en los discos de los servidores, que deberán estar disponibles de forma contínua durante largos (indefinidos) periodos de tiempo.

Existen diferentes técnicas disponibles para asegurar la consistencia de los datos alojados en los dispositivos de almacenamiento de nuestros servidores.

Todo el software citado a continuación formado de dos componentes: un controlador en el kernel, que tendremos que compilar (y que, salvo que se indique, viene de serie en el kernel 2.4.x y no tendremos que parchearlo), y una serie de utilidades en el espacio de usuario para modificar de alguna forma el funcionamiento del sistema (formatear particiones, etc.).

2.4.1 Gestión Avanzada de Discos

2.4.1.1 Redundant Array of Independent Disks (RAID)

Un subsistema RAID permitir eliminar SPF de los recursos de almacenamiento. RAID por software soporta un gran número de controladoras SCSI y ATA100 que ofrecen volúmenes de RAID por hardware.

El standard RAID consta de varios niveles, y en cada uno de ellos, el acceso a los discos y su contenido se organiza de manera diferente para conseguir mayor capacidad, que la que se obtiene con un único disco físico, mayor rapidez en el acceso a los datos, tolerancia a fallos, o alguna combinación de las anteriores.

Los distintos niveles de RAID son:

Modo lineal

No es un tipo de RAID, pero al elegir esta opción Linux permite unir dos o más discos haciendo coincidir el final de uno con el principio del siguiente, simulando tener un disco de una capacidad igual a la que suman los discos unidos. Este método no proporciona ningún tipo de protección contra fallos de cualquiera de los discos, ni tampoco acelera el acceso a ellos, con lo que no es una opción muy interesante en la práctica.

RAID 0 ó "stripe" (intercalado)

Este modo es similar al lineal, pero la información se va guardando en paralelo en ambos discos por bloques de un tamaño fijo. Tampoco añade seguridad, pero aumenta la velocidad de acceso a los dos dispositivos en paralelo. Los discos deben ser de aproximadamente el mismo tamaño y misma velocidad para obtener rendimientos óptimos.

RAID 1 ó "Mirroring (espejado)"

También conocido como disk mirroring. Une un par de discos idénticos para crear un único disco con la capacidad de uno de ellos, conteniendo ambos la

misma información. Este mecanismo no conlleva el entrelazado de los discos, pero se obtiene un aumento del rendimiento en lectura debido a que se pueden leer ambos discos independientemente. Este tipo de RAID es el que mayor rendimiento proporcionado dentro de los tipos redundantes, aunque a la vez es el método que más desaprovecha el espacio en los discos, ya que se utiliza realmente la mitad de la inversión en los mismos.

RAID 4:

En este modo se necesitan tres o más discos, en uno se guarda información de paridad y en los otros se almacenan los datos en paralelo. El tamaño del conjunto es de (N - 1)*T, siendo N el número total de discos activos y T el tamaño de los discos (o el del de menor tamaño, si no son iguales). Si falla un disco, la información se puede reconstruir gracias a los datos de paridad, si fallan dos, se pierde todo. Este modo RAID tiene un problema, que a pesar de escribir los datos en paralelo, como la información de paridad va siempre al mismo disco, éste se convierte en un cuello de botella, bajando el rendimiento del sistema.

RAID 5:

Utiliza tres o más discos, con o sin discos inactivos adicionales. Similar a RAID 4, pero la información de paridad se distribuye entre todos los discos, eliminando así el problema del cuello de botella con el disco de paridad. Si falla un disco, la información no se pierde gracias a la paridad, y el contenido del disco dañado se reconstruye en un disco inactivo.
Si fallan dos discos de forma simultánea, o si nos quedamos sin discos inactivos, la información se pierde. Tanto la velocidad de lectura como la de escritura aumentan, al realizarse en paralelo.

En el mercado existen dispositivos de almacenamiento de diversos fabricantes con configuraciones RAID, que para el sistema se comportan como un dispositivo normal (discos SCSI), pero internamente llevan varios discos y una controladora dedicada que accede a ellos según los niveles RAID.

Además algunos sistemas operativos son capaces de tomar varios dispositivos normales (discos IDE o SCSI) y realizar un RAID por software que, si bien resulta algo más lento que uno por hardware, ya que es el procesador del equipo y no la controladora dedicada quien tiene que tratar con la organización de los datos en los discos, también resultan mucho más baratos y flexibles que los dispositivos prefabricados.

2.4.1.2 LVM (Logical Volume Manager)

Es un subsistema que se encarga de la gestión avanzada de unidades de almacenamiento.
LVM ofrece una forma más potente y flexible de asignar en particiones el espacio físico de los discos duros. Con LVM agrupamos volúmenes físicos (PV, de Physical Volumes), que pueden ser cualquier dispositivo de bloques (particiones, discos completos o dispositivos /dev/md? del RAID por software) en grupos de volúmenes (VG, Volume Groups). Un VG consiste de uno o más PV, y se dividen en particiones virtuales al estilo de las tradicionales, denominadas volúmenes lógicos (LV, Logical Volumes).

Lo novedoso de esta tecnología es que, una vez configurados todos los volúmenes físicos y lógicos, podemos añadir o quitar en cualquier momento y en caliente (si el hardware y software lo permite) más volúmenes físicos a un grupo virtual, o más espacio a un volumen lógico. De esta forma, se elimina problema de tener que parar y reinstalar un sistema porque una partición se ha quedado pequeña y no se puede ampliar.

Igual que ocurre con RAID, jugando con estos modos de asignación podemos ganar en velocidad de lectura/escritura por el acceso a varios discos en paralelo.

Otra característica muy interesante de LVM es la posibilidad de crear “snapshots” (fotos) del sistema en un momento dado, algo muy útil a la hora de hacer una copia de seguridad.

2.4.2 Sistemas de Ficheros con journal

Una vez que se encuentra asignado el espacio en particiones tenemos que darle una estructura lógica para que acoja los directorios y ficheros. A esta estructura lógica se le conoce como sistema de ficheros.
Linux soporta una serie de sistemas de ficheros, algunos considerados “nativos” de este sistema y otros propios de otros sistemas operativos (vfat de Windows 9X/ME, NTFS de Windows NT/2000 o el HPFS de MAC). A continuación se enumeran los sistemas de ficheros más familiares y apropiados para la HA en Linux:

ext2

Es el sistema de ficheros por excelencia de Linux, y el que instala por defecto las distribuciones actuales. Ofrece funcionalidades estándar, soporta los archivos.

ext3

Es el clásico ext2 pero con el añadido de una partición de log, para llevar el journal. Hace journal de datos y metadatos. Por algo mas lento que ext2 en acceso pero mas rápido en tiempo de recuperación de consistencia. Lo realmente interesante es que hace journal de datos y metadatos.

Reiserfs

Es un sistema de ficheros creado desde 0 con la idea de sacar partido a árboles B, hashes y técnicas de journal actuales. Es un sistema de ficheros realmente rápido en lo que a Linux se refiere. Sólo hace journaling de datos. Actualmente sus creadores están desarrollando Reiser4 bajo el patrocinio de la DARPA.

JFS

IBM lo ha puesto bajo GPL y ha sacado su primera versión con calidad de producción. De momento desarrollan con bastante independencia, de versiones actuales del kernel, ya que sacan parches para versiones especificas.

XFS

Silicon ha sacado releases con calidad de producción bajo GPL. Casi tan rápido como reiserfs. Esta muy bien integrado con Linux, teniendo en consideración otros subsistemas como puede ser quota o NFS. Sacan parches cada una o dos versiones del kernel. Una opción muy a considerar, en un futuro.

2.5 Distribución de Datos

La alta disponibilidad implica que los datos que tengan que servir o procesar deben estar disponibles para todos y cada uno de los servidores, de forma que para el usuario el cluster se comporte como un único ordenador, en el que ellos copian en un único lugar los ficheros, y el software de control del cluster internamente se encargue de hacer llegar una copia a cada uno de los servidores que lo componen.
Existen dos estrategias: la replicación física de archivos, en la que cada servidor tendrá una copia de todos los datos en su disco duro; y la distribución de los datos mediante sistemas de archivos distribuidos, en los que un servidor de ficheros y el resto de equipos del cluster accederán a sus contenidos por la red.

2.5.1 Replicación de Archivos

Esta es la alternativa más “primitiva” para la distribución del contenido a servir a todos los equipos del cluster, es la replicación (automática o manual) de los ficheros en todos los ordenadores.

Aquí veremos un novedoso protocolo que optimizará en gran medida la cantidad de datos a transmitir por la red y, en consecuencia, el tiempo necesario para realizar la sincronización.

2.5.1.1 Rsync

Es un programa para la sincronización remota de archivos de forma eficiente, transfiriendo solamente las partes que son diferentes (que pueden ser un porcentaje muy pequeño), y además las puede comprimir (-z). Rsync puede trabajar como cliente y como servidor

Con rsync podemos:

- Copiar o sincronizar ficheros, directorios o sistemas de archivos enteros, manteniendo enlaces, permisos, fechas, etc.
- Redireccionar todo el tráfico a través de ssh para cifrarlo.
- Permitir un acceso “anónimo” para que terceras personas puedan hacer mirror de nuestras páginas.
- Copiar por la red únicamente las diferencias entre los ficheros.

El algoritmo rsync

Al algoritmo rsync funciona a grandes rasgos de la siguiente forma:
El equipo receptor ('A') divide el fichero en bloques de un tamaño fijo que no se solapan entre sí.

Se dispone de dos algoritmos de cálculo de CRC, uno muy rápido (‘X’) pero no exacto, aún que asegura que nunca dará un falso negativo (un bloque con CRC correcto siempre será evaluado como correcto; uno incorrecto puede que sea evaluado como correcto).

Además, este algoritmo tiene la característica de que el resultado del bloque x+1 se puede calcular rápidamente a partir del resultado del bloque x.

Otro más lento (‘Y'), pero que sí que es capaz de discriminar siempre si el CRC es correcto o no. Se calculan los valores de ambos algoritmos sobre los bloques del fichero, y se envían al otro equipo.

El equipo emisor ('B') busca en su fichero bloques del tamaño fijado para los que coincida el algoritmo ‘X’: Si el CRC difiere en el fichero local y el remoto, no hace falta retransmitir este bloque, si coincide, se analiza con el algoritmo ‘Y’.

Si con el algoritmo ‘Y’ también coincide, entonces el bloque ha cambiado y habrá que transmitirlo. Se envía al otro equipo la información precisa para reconstruir el fichero, bien como una referencia a otro bloque del fichero o como datos puros.

De esta forma, se consigue disminuir en gran medida la cantidad de datos a transmitir entre las dos máquinas, algo muy a tener en cuenta si la conexión entre los equipos es lenta o si el número de ficheros a sincronizar es grande.

2.5.1.2 NFS

NFS permite compartir datos entre varios ordenadores de una forma sencilla. Por ejemplo, un usuario validado en una red no necesitará hacer login a un ordenador específico: vía NFS, accederá a su directorio personal (llamado exportado) en la máquina en la que esté trabajando.
NFS no es un protocolo demasiado eficiente y es muy lento para conexiones mediante módem. Está diseñado para redes locales, siendo muy flexible.

La forma de trabajar de NFS es la siguiente. Un cliente intenta montar (conectar a su árbol de directorios) un directorio desde un host remoto en un directorio local de la misma forma que si fuera un dispositivo físico. Sin embargo la sintaxis empleada para montar el directorio remoto es diferente:

```
[root@nodo1]# mount -t nfs hostname:/dirremoto /dirlocal
```

Los 4 servicios que permiten funcionar a NFS son:

Protocolo	Descripción	Demonio
Nfs	Este protocolo permite crear, buscar, leer o escribir ficheros, también maneja autentificación y estadísticas de ficheros.	Nfsd
Mountd	Montar sistemas exportados para acceder a ellos con nfs. El servidor recibe peticiones como mount y umount debiendo mantener información sobre los sistemas de ficheros exportados.	Mountd
Nsm (Network Status Monitor)	Es utilizado para monitorizar los nodos de la red y así conocer el estado de una máquina (cliente o servidor).	Statu
Nlm (Network Lock Manager)	Para impedir modificaciones de los datos por varios clientes al mismo tiempo, este protocolo maneja un sistema de bloqueo.	lockd

Samba

Samba proporciona una interoperabilidad transparente para el usuario con sistemas Windows. Aun que los sistemas Unix sean los más implantados en el terreno de los servidores de Internet, no cabe duda que en cuanto a sistemas “de escritorio”, Windows es el más extendido, y la mayoría los usuarios, que tengan que acceder a nuestro servidor lo harán desde Windows.

El protocolo SMB es usado por Microsoft Windows 3.11, 9x/ME y NT/2000 para compartir discos e impresoras. Usando el paquete de herramientas Samba, las máquinas UNIX (incluyendo Linux) pueden compartir discos e impresoras con servidores Windows.

Existen cuatro cosas que podemos hacer con Samba:
1. Compartir una unidad de Linux con máquinas Windows.
2. Compartir una unidad de Windows con máquinas Linux.

3. Compartir una impresora de Linux con máquinas Windows.
4. Compartir una impresora de Windows con máquinas Linux.

Coda

Coda es un sistema de archivos distribuido avanzado, con sus orígenes en el AFS2.
Coda cuenta con bastantes características muy deseables para un sistema distribuido (especialmente para un cluster) que no se encuentran en ningún otro sistema similar, entre otras:

- El cliente es capaz de funcionar sin problemas desconectado del servidor. Cuando se restablece la comunicación, los sistemas del cliente y el servidor se sincronizan de forma automática.
- Alto rendimiento proporcionado por una caché local en la máquina cliente: la primera vez que se accede a un fichero se guarda en el disco duro local, y los siguientes accesos se realizan en local tras comprobar que la copia es válida (la del servidor no ha sido modificada).
- Replicación automática de servidores. Coda proporciona los mecanismos necesarios para realizar réplicas automáticas entre servidores, y para que los clientes puedan acceder a uno u otro de forma transparente para el usuario si alguno cae.
- Buena escalabilidad.

Todas estas características hacen de CODA un sistema distribuido idóneo para un cluster de alta disponibilidad.

2.6 Gestión de Monitorización

La monitorización de servicios es un aspecto muy importante en el clustering de alta disponibilidad: ya que si algún servidor falla, es fundamental advertirlo de alguna forma y efectuar las acciones pertinentes (eliminarlo de la lista de servidores activos y hacer que algún otro servidor tome el lugar de este).

2.6.1 Daemontools y ucspi-tcp

Daemontools no es un programa de monitorización de servicios en sí, Daemontools controla las conexiones a una serie de servicios esperando conexiones en una serie de puertos, y lanza los servicios asociados cuando se necesiten.
Por otro lado, ucspi -tcp lanza un servicio cuando se le indique, pero mirando la IP del cliente y comparándola contra una lista de IPs permitidas/prohibidas, controlando un número máximo de conexiones simultáneas (para evitar ataques tipo DoS), etc.

Además, daemontools comprueba si un servidor se ha "caído" y se encarga de volverlo a lanzar, una vez por segundo para no sobrecargar la máquina en caso de que haya algo mal en la configuración, de esta manera tendremos controlado el caso más simple de monitorización.

2.6.2 mon

mon realiza la monitorización a nivel de servicio. Se definen una serie de pequeños scripts que se conectarán a cada puerto que se desea monitorizar y lanzarán una pregunta de la que sepamos la respuesta que nos dará el servidor si no falla nada, y así averiguaremos el estado del servicio.

Este paquete está hecho prácticamente en perl, por lo que se debe instalar todos los módulos de perl que se requieran para correr el programa.

El principal uso de mon es el de monitorizar servicios remotos en las máquinas que conforman el cluster, para quitarlas de éste en caso de que alguno falle, aunque, también se puede utilizar sobre la máquina local para detectar si algún servicio funciona mal y reiniciarlo si fuera necesario.

En el fichero de configuración se definen que máquinas y que servicios se desean monitorear, indicando en cada caso el intervalo a utilizar para la monitorización y las acciones a tomar en caso de que el resultado de la

monitorización sea negativo. En este fichero encontramos las siguientes secciones:

• La primera es la sección Hostgroups, que aporta los grupos a los que se realizará la monitorización.

• La segunda es la sección de vistas, que se compone de una o más vistas sobre los grupos de trabajo que contiene las especificaciones de la monitorización y la actuación que se hará acorde con cada evento. En las zonas de vistas se pueden separar en:

- El servicio a monitorizar, pueden contenerse uno o más servicios en cada vista, de manera que se especifica en ellos.
- El intervalo de resolución en el que se realizaran los sondeos de las monitorizaciones.
- Interrupciones o señales que se utilizarán.
- El programa monitor que se utilizará para cada servicio.
- La dependencia entre servicios y el comportamiento en caso de dependencias
- El campo periodo que especifica entre qué fechas se realizará la monitorización y de qué manera se procederá en caso de ciertos eventos. control de las alertas

Tanto los monitores como las acciones son programas que se pueden crear en cualquier lenguaje.

Un monitor, por regla general, se conectará a un puerto, lanzará una petición y leerá una respuesta; una alarma puede enviar un mail, un mensaje SMS al móvil, o tal vez tratar de solucionar el problema.

2.6.3 Heartbeat

Heartbeat se utiliza para monitorizar y significa "latido de corazón", lo que hace es periódicamente enviar un paquete, que si no llegara indicaría que un servidor no está disponible por lo tanto se sabe que el servidor ha caído y se toman las medidas necesarias.

Cuando un ordenador deja de hacer heartbeats y se considera muerto, se hace una transición en el cluster. La mayoría de los mensajes de manejo del cluster que no son heartbeats se realizan durante estas transiciones.

Los mensajes de Heartbeat se envían por todas las líneas de comunicación a la vez, así si una línea de apoyo cae, se avisará de ese problema antes de que la línea principal caiga y no haya una línea secundaria para continuar el servicio.

Heartbeat tiene el problema que si no tiene una línea dedicada, aunque ésta sea una línea serie, al tener un tráfico que aunque pequeño es constante, suele dar muchas colisiones con otros tráficos que puedan ir por la misma red.

Heartbeat proporciona un mecanismo para que dos servidores controlen su estado mutuamente a través de varios accesos: red Ethernet, cable serie, etc. De esta forma, al tener varias conexiones y realizar la comprobación por todas ellas, no se incurrirán en errores como creer que un servidor ha caído cuando realmente lo que existe es un problema con la red.

La comprobación de estado se realiza a nivel de aplicación, ya que los servicios heartbeat en cada máquina se comunican entre sí mediante un protocolo propio, que además va cifrado para asegurar la identidad de cada máquina.

heartbeat también puede controlar “cuelgues” en la propia máquina, programando algún dispositivo watchdog (hardware o software en el kernel) para reiniciar la máquina de forma automática. El funcionamiento del watchdog es el siguiente: este dispositivo se programa para que tenga que recibir una entrada (algún texto, lo que sea) cada x tiempo. En caso de fallar ésta entrada varias veces seguidas, el watchdog se encargará de reiniciar la máquina de forma automática.

heartbeat se puede programar para que él mismo se encargue de conectar con el watchdog con la frecuencia programada para que no reinicie la máquina, y

para que deje de hacerlo si detecta algún problema grave en el equipo (o si se cuelga, con lo que dejaría de enviar estas señales y el watchdog reiniciaría la máquina).

Otra característica importante de heartbeat es que puede “adueñarse” de la IP de otra máquina (la que está monitorizando) mediante la técnica conocida como ARP IP spoofing: cuando detecta que la otra máquina ha caído, comienza a enviar tramas ARP anunciando sus direcciones IP y MAC, con lo que el resto de equipos, routers, y demás dispositivos de red asociarán a partir de ese momento la IP indicada con la dirección MAC de la tarjeta del servidor.

2.7 Cluster de Alta Disponibilidad

Un cluster de Alta Disponibilidad (High Availability) consiste en varias máquinas que comparten los discos duros y se monitorizan entre sí constantemente. Su utilidad se basa en la necesidad de tener un sistema capaz de reponerse de un fallo de hardware y de la necesidad de abaratar los costes de los sistemas redundantes y tolerantes a fallos. Cuando uno de los equipos cae, los demás migran sus procesos hacia ellos mismos para que no haya pérdidas de datos y de disponibilidad del conjunto. A su vez, se encargan de rearrancar al ordenador caído y, tras lograrlo, le devuelven su carga correspondiente para volver a la normalidad.

Los sistemas de alta disponibilidad están basados en la replicación de elementos, mucho más baratos que un sólo elemento tolerante a fallos, es decir si se habla de replicar servidores, estamos hablando de un clúster de alta disponibilidad.

Lo que se busca con la replicación de máquinas es evitar los puntos únicos de fallo, del inglés SPF (Single Point of Failure), que serían aquellas máquinas imprescindibles para el correcto funcionamiento del servicio que queremos dar: si únicamente tenemos una instancia de cada máquina de este tipo, se convierte en un SPF y ante cualquier fallo en este equipo, todo el cluster queda inutilizado. La teoría sobre este tipo de clusters gira en torno a estos SPF y cómo evitarlos, mediante redundancia hardware y el software apropiado para controlar el

correcto funcionamiento de todos los equipos y, en caso negativo, hacer que una máquina de respaldo suplante a la que acaba de fallar.

Los clusters de alta disponibilidad necesitan de un amplio abanico de componentes que consideren diversos factores, entre otros:

- Control de miembros del cluster.
- Servicios de comunicaciones.
- Control y gestión del clustering, flujo de datos.
- Gestión y monitorización de recursos.
- Compartición o replicación del almacenamiento:
  - Compartición:
    - Discos SCSI externos.
    - Sistemas NAS.
    - Sistemas de ficheros compartidos (NFS, SMB, Coda).
  - Replicación:
    - Propio de la aplicación (DNS, NIS, etc.)
    - ftp, rsync, etc.

Un buen cluster de Alta Disponibilidad debe proveer fiabilidad, disponibilidad y servicio (RAS: Reliability, Availability, Serviceability).

- **Confiabilidad** (Reliability)

  El sistema debe ser confiable al realizar las tareas para la cual ha sido configurado, es decir debe ser capaz de coordinar el sistema cuando éste está distribuido entre nodos y además debe de integrar todo el software de manera que funcione entre si de manera confiable. En general se trata de que el sistema pueda operar sin ningún tipo de caída o fallo de servicio.

- **Disponibilidad** (Availability)

La disponibilidad es la base de este tipo de clusters, ya que indica el porcentaje de tiempo que el sistema esta disponible en su funcionalidad hacia los usuarios.

- **Servicio** (Serviceability)

  Hace referencia cuán fácil es controlar los servicios del sistema y que servicios se proveen, incluyendo todos los componentes del sistema.

2.8 Técnicas para proveer de disponibilidad

2.8.1 Técnicas basadas en redundancia

Estas técnicas consisten en reducir el tiempo de fallo o caída de funcionamiento, las cuales se basan fundamentalmente en efectuar algún tipo de redundancia sobre los dispositivos críticos. Para saber cuales son estos dispositivos debemos aplicar nuestro conocimiento sobre el sistema y sentido común. Estas técnicas permiten que cuando se presenta la caída de uno de estos recursos, otro tome la funcionalidad o configuración y actuación del otro, cuando esto ocurre, el recurso master puede ser reparado mientras que el recurso backup toma el control.

En algunos casos el dispositivo master reparado pasa a ser considerado backup del sistema y el antiguo backup pasa a ser considerado como master. Entre los tipos de redundancia que pueden presentar los sistemas, tenemos:

- **Redundancia aislada.**

  Existen 2 copias para dar una funcionalidad o servicio. Por un lado tenemos la copia maestro y por otro lado la copia esclavo. Cuando cae el servicio o recurso, la copia redundante pasa a ser la utilizada, de esta manera el tiempo de caída es mínimo o inexistente. Los recursos de los que hablamos pueden ser de cualquier tipo.

  Entre las ventajas que posee esta técnica, tenemos:

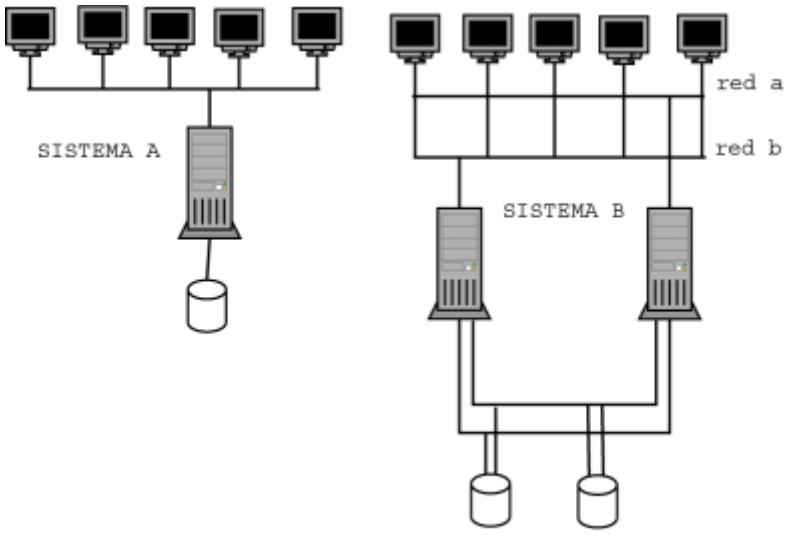
No existen puntos críticos de fallo en el sistema que bajen la confiabilidad del mismo.

Los componentes que han fallado pueden ser reparados sin que esto cause en el sistema una parada.

Cada componente del sistema puede comprobar de manera periódica si se ha producido algún tipo de fallo en los sistemas de backup.

- **N-Redundancia.**

  Es igual a la Redundancia Aislada, pero se tienen N equipos para ofrecer un mismo servicio, con lo cual presenta más tolerancia a fallos.



Para el caso de la figura se necesita replicar:

- LAN
- LAN para los servidores
- Los servidores
- El bus SCSI de acceso a discos duros
- Discos duros

Como se muestra en la figura, la replicación proporciona rutas alternativas, discos alternativos y en general recursos alternativos, y es aplicada sobre todos aquellos recursos que se consideren críticos en una red.

Otro punto importante que se debe tomar en cuenta al momento de la instalación de los dispositivos redundantes es la configuración o el modelo de funcionamiento de éstos. Estos depende de cómo sea implementado el software y aplicado al hardware, y define el modo de comportamiento de como es la redundancia en sí misma. Esta redundancia puede ser del tipo.

- **Hot Standby**

  Este tipo de configuración, hace referencia a cuando cae el nodo master, un nodo backup toma el control de sus operaciones. El inconveniente en esta es el doble gasto sobre recursos que en un principio se están desperdiciando. El servidor de backup simplemente monitoriza sus conexiones, para asegurar que cuando el nodo maestro caiga, él tomará el control de las operaciones. Exceptuando estas operaciones el nodo backup no hace nada, por lo tanto, el costo que se invirtió en este equipo es probable que se esté desperdiciando, por lo menos hasta el momento en el que entre en acción (probablemente en una fecha crítica).

- **Toma de cargos mutua**

  La toma de cargos mutua, es una configuración que soluciona la desventaja del Hot Stanby. Mientras el nodo principal se ocupa de proveer el servicio, el nodo de backup hace las mismas operaciones que en la configuración anterior y además puede efectuar otro tipo de operaciones. De este modo, la capacidad de este nodo se está aprovechando más y el costo del sistema se ve recompensado con un equipo más que se utiliza para efectuar trabajo útil. El problema está cuando el nodo maestro cae. En este caso, el comportamiento del backup puede tomar dos vías.

  - Primero mantener sus actuales trabajos y tomar los trabajos que el maestro ha dejado sin hacer. Esta manera de comportarse presenta una baja de rendimiento del sistema crítico, pero hace que esté disponible.

  - Segundo, dejar estos trabajos postergados hasta que el antiguo master sea reparado y cuando este tome las tareas de backup, que continúe con el trabajo que en un principio estaba efectuando él antes de tomar el control. Esta es una solución

mucho más difícil de implementar, pero presenta mejor costo y rendimiento que la anterior.

- **Tolerante a fallos**

  Los sistemas redundantes a fallos se basan en la N-Redundancia. Si se tienen N equipos y caen N-1 el sistema sigue en funcionamiento. Este sistema se puede complementar con la toma de cargos mutua para no perder rendimiento ni elevar el costo del sistema, sin embargo configurar un sistema de este tipo es bastante complejo a medida que aumenta N.

2.8.2 Técnicas basadas en reparación

Estas técnicas se basan en reducir el tiempo de reparación. Estas técnicas se componen de scripts o programas que detectan donde falló el sistema y tratan de recuperarlo sin necesidad de un técnico especializado, es decir, son técnicas de automatización de tareas basadas en sistemas expertos. Se pueden dividir en dos tipos de acciones que realizan, las cuales pueden ser independientes o dependientes entre si:

- **Diagnóstico.**

  La parte de diagnóstico simplemente trata de conocer las posibles causas que han provocado el error y principalmente localizar el error. Una técnica muy utilizada en este campo es una especie de piggybacking aplicada a los pulsos o latidos entre ambos nodos. En esta técnica, se envía junto con el latido o pulso, la suficiente información como para prever cual será el estado de los componentes en próximos tiempos o incluso actualmente.

- **Reparación.**

  Son técnicas mediante las cuales a través de sistemas expertos o cualquier otro tipo de actuación el sistema caído puede llegar a ser restaurado desde un sistema copia. En la mayoría de los casos basa su funcionamiento en

puntos de comprobación o checkpoints que se efectúan sobre el sistema cada cierto tiempo, de manera que el servidor caído es restaurado al último checkpoint existente. Los puntos críticos son:

- Aislar los componentes que fallan y sustituirlos o repararlos. Estos pueden ser localizados mediante programas que implementen sistemas de comprobación o sistemas expertos.
- Recuperación mediante puntos de comprobación o puntos de restauración.
- Acoplamiento al cluster actual tomando las acciones que tenía el nodo backup e informando al nodo master de que ya existe un nuevo backup en el sistema.

Los puntos de comprobación son importantes ya que introducen un factor de sobrecarga en el sistema y al mismo tiempo son un factor crítico a la hora de efectuar restauraciones del mismo. Un sistema confiable al máximo debería guardar el estado de todos los programas que están corriendo y comunicárselos a su backup en tiempo real para que de este modo la caída de uno guardase el estado del complementario. Serían sistemas simétricos. Este tipo de sistemas solamente son implementados en hardware, ya que exigen medios de comunicación demasiado rápidos aparte de la sobrecarga al procesador que genera el estar controlando este tipo de labores.

Los clusters implementan a un nivel mucho menos eficiente este tipo de checkpoints, y la eficiencia depende generalmente de la capacidad de los procesadores y de la capacidad de la red que une maestro y esclavos (backups), debido a esto, los clusters solamente guardan configuraciones y servicios activos.

2.9 Dinámica de Alta Disponibilidad

La dinámica de Alta Disponibilidad hace referencia a todas las reconfiguraciones del clúster que garanticen la máxima disponibilidad del servicio. Esta dinámica esta orientada a los nodos integrantes del clúster y la forma en la cual el clúster responde.

A esta dinámica se la llama Failover, este término es utilizado para indicar cuando un nodo debe asumir la responsabilidad de otro nodo, importar sus recursos y levantar el servicio de datos. El failover es una situación excepcional, para cual la alta disponibilidad ha sido concebida, el fallo de un nodo.

Existen soluciones concretas, que pueden ser scripts escritos en cualquier lenguaje que se encargan de adquirir datos del sistema, y en el caso de que el resultado no sea el adecuado, ejecutar las sentencias que solucionen el problema.

Takeover es un failover automático, se produce cuando un nodo nota un fallo en el servicio. Para ello debe haber cierta monitorización con respecto al mismo. El nodo que se declara fallido, es forzado a ceder el servicio, recursos, o simplemente ser eliminado.

Switchover o Giveaway es un failover manual, consiste en ceder los recursos de un servicio y este mismo, a otro nodo del clúster, mientras se realizan ciertas tareas administrativas. A este procedimiento se le denomina Node outage.

Un Splitbrain (división de cerebros) es un caso especial de failover, en el cual falla el mecanismo de comunicación y gestión de un clúster de dos nodos. Es una situación en la cual cada nodo cree que es el único activo, y como no puede saber el estado de su nodo compañero, tomará acciones en consecuencia, forzando un takeover. Esta situación es peligrosa, los dos nodos intentaran apropiarse de todos los recursos, incluyendo el servicio. El peligro aumenta sobre todo cuando tenemos recursos delicados, como recursos de almacenamiento, ya que cada nodo podría tomar y escribir por su cuenta, y quebrar a integridad de datos. Para evitar este problema, cada nodo debe actuar de una forma prudente, y utilizar los recursos compartidos como señal de que se esta vivo. La forma de proceder de cada nodo es similar, ya que cada uno cree que su compañero ha desaparecido.

Después de que un nodo aprecia este problema, tiene indicado que reserve un recurso llamado quorum. Un recurso quorum, es un recurso compartido, que se ha preestablecido en ambos nodos. Este recurso es un recurso exclusivo, sólo un

nodo del cluster puede reservarlo, por lo tanto, el nodo que llegue tarde a la reserva del recurso, entiende que debe abandonar el clúster y ceder todos sus recursos.

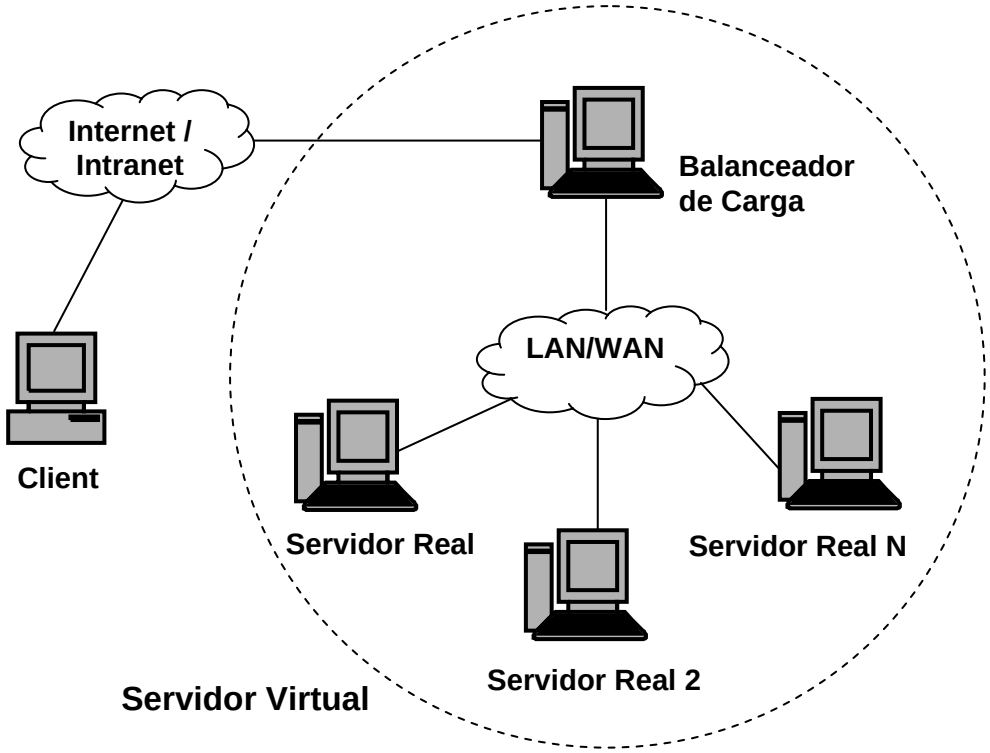
El quórum es utilizado como método de decisión. Otra forma de evitar esta situación, un poco mas violenta, es que un nodo elimine a su compañero, el primero que apague a su compañero se queda con todos los recursos. Es un mecanismo muy brusco, pero muy eficaz. Para este caso existen tarjetas especializadas en esta tarea. Pensando en un servicio de datos cualquiera, se puede pensar en una serie de recursos que va a utilizar.

2.10 Linux Virtual Server (LVS)

El proyecto Linux Virtual Server nos provee con la información y todos los programas necesarios para montar un “servidor virtual” fácilmente escalable sobre un cluster de máquinas Linux.

Linux Virtual Server se basa únicamente en PCs corriendo Linux y no necesita de ningún router/firewall/balanceador hardware, ni ningún software propietario de terceras personas.

2.10.1 Visión general de LVS



El usuario se conecta a través de Internet a la dirección pública del cluster, que está asignada al “balanceador de carga”. Este equipo está conectado a través de una LAN o una WAN con el resto de equipos del cluster y se encarga de dirigir cada una de las peticiones al servidor que se encuentre con menor carga para atenderla. Según la configuración por la que optemos, la respuesta será enviada por el servidor real directamente al cliente o será de nuevo redirigida a través del balanceador de carga.

La escalabilidad se conseguirá fácilmente añadiendo más equipos a nuestra LAN, aumentando así el número de servidores reales en el cluster. También aumentando la alta disponibilidad, ya que al tener varios servidores, cuando uno falle el resto asumirán la carga del caído.

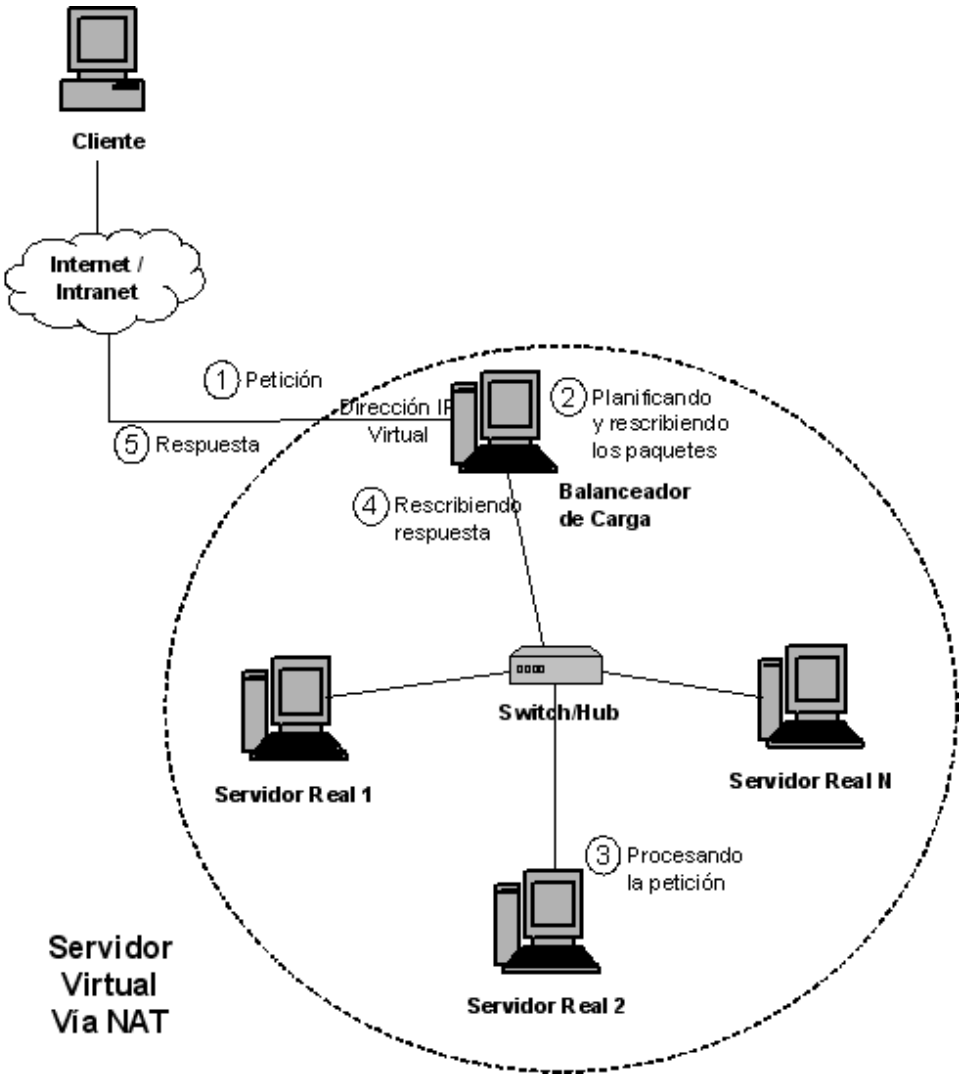
2.10.2 Modos de balanceado de carga en LVS

LVS permite realizar el reenvío de paquetes a los servidores reales de tres formas, que se explican a continuación:

2.10.2.1 Balanceado por NAT (VS-NAT)

Este tipo de balanceado aprovecha la posibilidad del kernel de Linux de funcionar como un router con NAT (Network Address Translation), que permite modificar las direcciones de origen/destino de los paquetes TCP/IP que lo atraviesen. La única dirección real del cluster será la del balanceador; cuando le llegue un paquete modificará la dirección de destino para dirigirlo a uno de los servidores y la de origen para que le sea devuelto a él, y lo reenviará a la red privada. Cuando el servidor real lo procesa, se lo envía al balanceador y éste realiza el cambio de direcciones: poniendo su IP como dirección de origen del paquete con la respuesta, y como dirección de destino la del cliente que originó la petición.

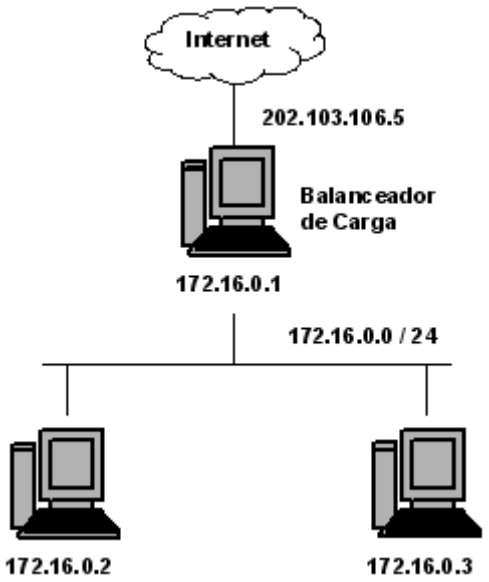
En el siguiente gráfico, se puede ver todo el proceso.



Donde cada punto indica que:[4]

1. El cliente realiza una petición de servicio, a la IP pública del cluster (la del balanceador de carga).
2. El balanceador planifica a qué servidor real va a enviar la petición, rescribe las cabeceras de las tramas TCP/IP y se las envía al servidor.
3. El servidor recibe la petición, la procesa, genera la respuesta y se la envía al balanceador de carga.
4. El balanceador rescribe de nuevo las cabeceras de las tramas TCP/IP con la respuesta del servidor, y se las envía de vuelta al cliente.
5. La respuesta llega al cliente, como si la hubiera generado la IP pública del cluster.

Visto de una forma más "física", la implementación del cluster sería de la siguiente manera:



La única IP pública del cluster sería la 202.103.106.5, que sería la IP que asociaríamos en el servidor de DNS al dominio de nuestro cluster y a la que irían dirigidas todas las peticiones de los clientes. El resto de direcciones son de la red privada.

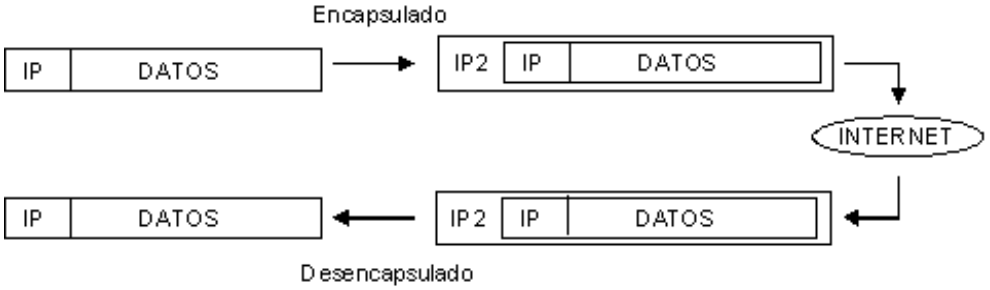
Una gran ventaja de esta técnica es que los servidores que componen el cluster, pueden ejecutar cualquier Sistema Operativo que soporte TCP/IP, ya que toda la

manipulación de direcciones y gestión del cluster se realiza en el balanceador, sin ser necesario modificar de ningún modo el resto de servidores. Además, sólo se necesita una IP real (la que asignaremos al balanceador). El resto de servidores tendrán IPs privadas de una red local.

Sin embargo, el problema de este método es que el balanceador se puede llegar a convertir en un cuello de botella, ya que todo el tráfico de entrada y salida al cluster pasa por él y además tiene que rescribir todos los paquetes TCP/IP. Afectando a la escalabilidad del cluster ya que dependerá de estos factores.

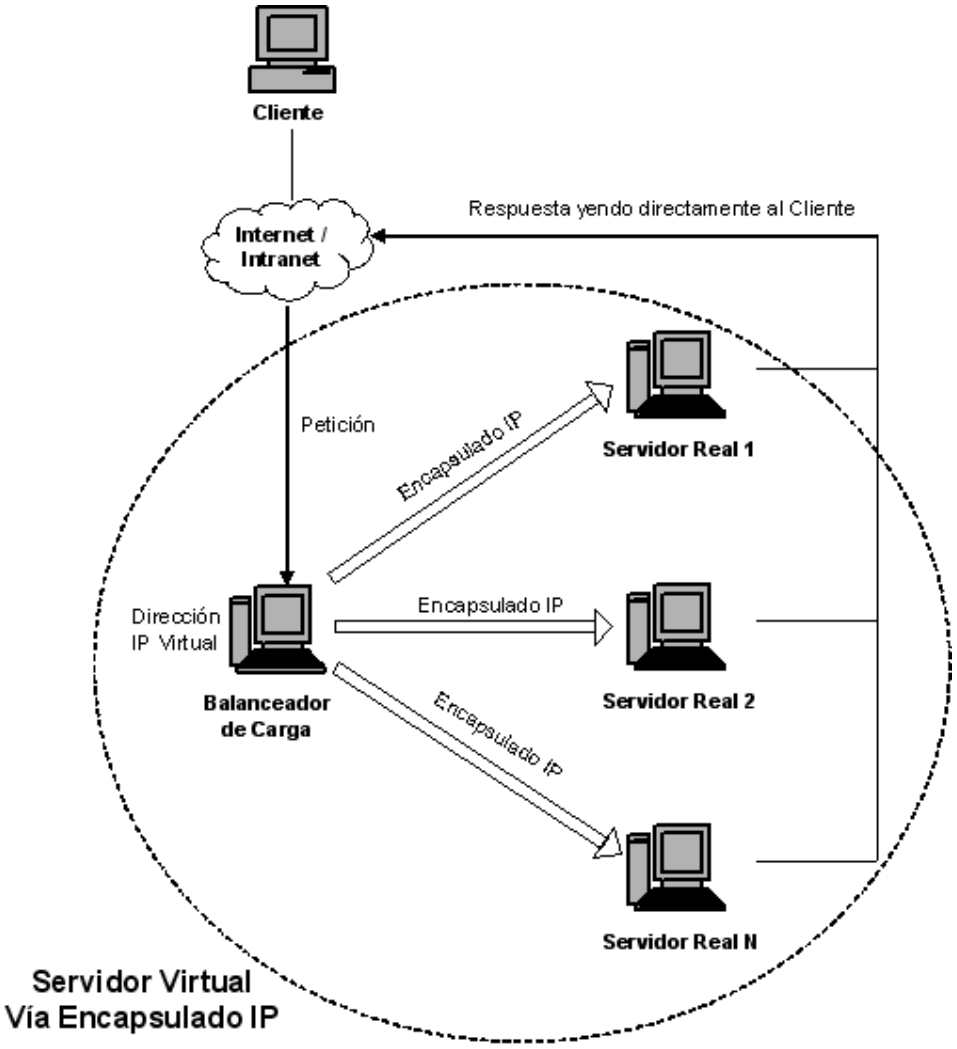
2.10.2.2 Balanceado por encapsulado IP (VS-Tun)

Este método permite escalar hasta un mayor número de servidores, 100 o más, pero todos ellos deberán ser capaces de soportar el encapsulado IP (IP tunneling). El encapsulado IP consiste en dirigir una trama TCP/IP, con sus direcciones de origen y destino, dentro de otra trama con direcciones distintas, y una vez que la trama más externa llegue a su destino, "desencapsular" la trama original y reenrutarla desde allí. El problema de esta técnica es que no todos los Sistemas Operativos la soportan. El encapsulado IP funciona básicamente de la siguiente manera:



Al utilizar este método de balanceado, todos los servidores deben tener configurada la IP pública del servidor en alguna interfaz (aún que sea virtual), y además necesitan IPs públicas si se va a distribuir los equipos en una WAN. Ya que al realizar la comunicación entre el balanceador y los servidores por medio de encapsulado IP, es posible distribuir los servidores reales a lo largo de una red de área amplia (WAN).

El esquema de LVS con encapsulado IP se muestra en la siguiente figura:



El balanceador recibe todas las conexiones de entrada al cluster, y decide a qué servidor enviarlas. Para hacer esto, utiliza el encapsulado IP para enviar cada trama que recibe a la IP del servidor que vaya a encargarse de ella. Cuando el servidor elegido recibe el paquete, lo desencapsula y al tener configurada la IP pública del cluster como propia, acepta la trama original y se encarga de realizar la petición que contiene la trama y enruta directamente la respuesta hacia el cliente sin necesidad de pasar de nuevo por el balanceador.
Con esta técnica se evita que el balanceador sea un cuello de botella haciendo que sólo los paquetes de entrada al cluster pasen a través de él.

El mayor problema de este método es que todos los servidores deben entender el encapsulado IP.

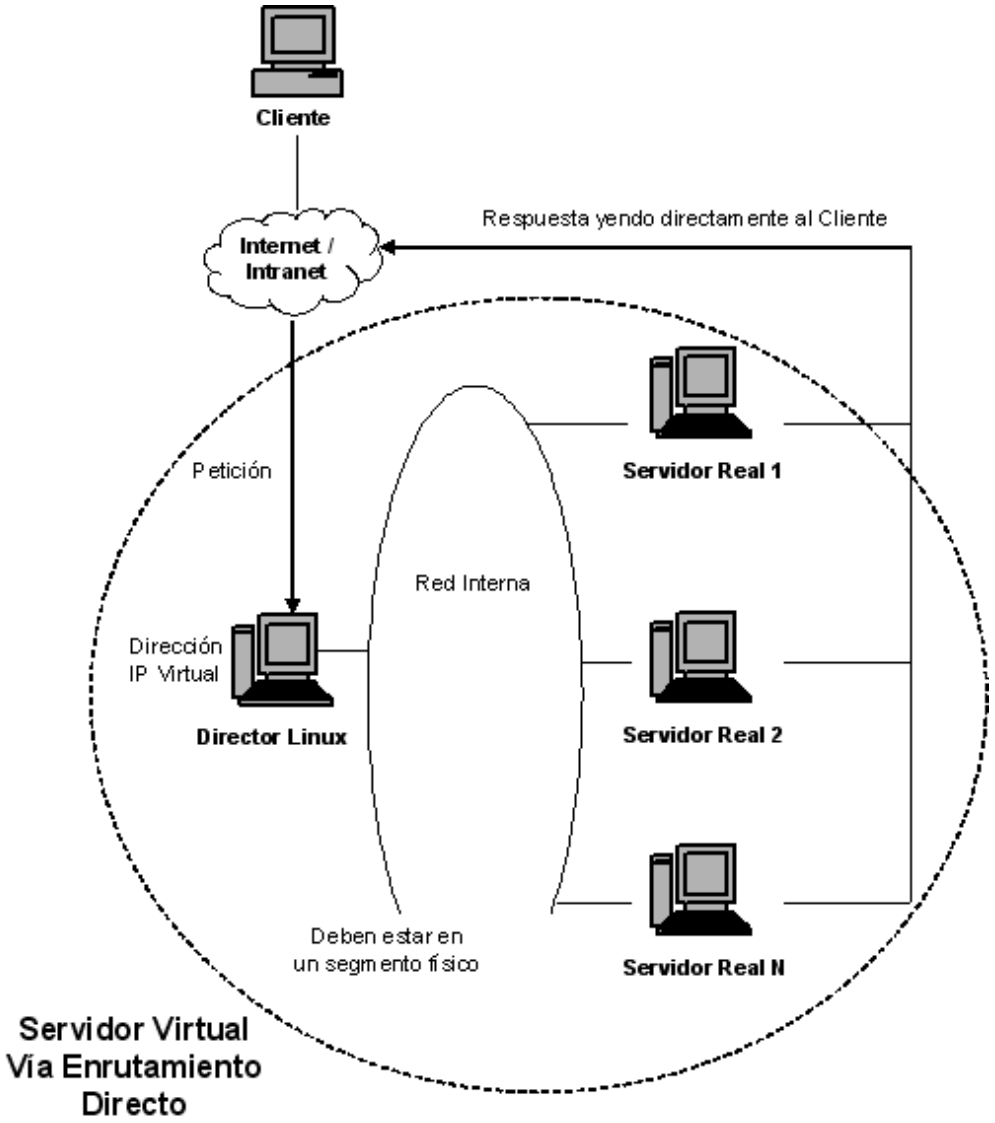
Además, con este método todos los servidores del cluster necesitan tener IPs públicas, lo que sería una desventaja, por el costo de la adquisición de éstas, aún que esto llevaría a que los servidores estén dispersos en una red de área amplia. Al poder separar geográficamente los servidores se añade un punto más a la alta disponibilidad del cluster, ya que ante un fallo general en la localización de un cluster centralizado se inutilizaría todo el cluster, mientras que si los equipos están dispersos esto es más difícil que ocurra.

2.10.2.3 Balanceado por enrutamiento directo (VS-DR)

Este tercer método de balanceo requiere que todos los equipos tengan configurado una interfaz con la IP pública del cluster: el balanceador, la tendrá en su acceso a Internet y será el punto de entrada al cluster; el resto de equipos estarán conectados al balanceador en la misma red física y en la interfaz conectado a esta red tendrán configurada la IP pública del cluster, pero configurando la interfaz para que no responda a comandos ARP para no interferir con otros protocolos (todos los equipos responderían por la misma IP con distintas MACs).

Dando las siguientes ventajas: Impone menos sobrecarga al equipo balanceador, ya que no tiene que rescribir los paquetes (caso NAT), ni encapsularlos (caso encapsulamiento IP). Además, el balanceador no seria un cuello de botella, ya que únicamente pasará a través de éste el tráfico de los clientes hacia el cluster, mientras que el tráfico de salida lo dirigirán directamente los servidores a cada cliente.

Cuando llega una petición al balanceador, éste decide a qué servidor enviarla, y redirige el paquete a nivel de enlace (ej. Ethernet) a la dirección MAC del servidor elegido. Cuando llega al servidor con la MAC de destino y se analiza hasta el nivel de red (TCP/IP), como el servidor también tiene configurada la IP pública del cluster, acepta sin inconvenientes el paquete y genera la respuesta, enviándola directamente al cliente.



Resumen de los métodos de balanceado

En la siguiente tabla se resumen las características principales de los tres métodos de direccionamiento que puede utilizar el balanceador de carga de Linux Virtual Server:

	NAT	Encapsulamiento IP	Enrutamiento Directo
Servidor	Cualquiera	Necesita encapsulamiento	Dispositivo no-ARP
Red de Servidores	Red Privada	LAN / WAN	LAN
Escalabilidad	Baja (10 – 20)	Alta	Alta
Salida hacia Internet	Balanceador	Router	Router

2.10.3 Planificación del balanceo de carga

Al momento de configurar el balanceador se puede elegir entre una serie de algoritmos para ver cómo se distribuirá la carga entre los servidores y cómo se elegirá el servidor al que se envía cada petición. Linux Virtual Server permite utilizar los siguientes algoritmos:

2.10.3.1 Round Robin

La cola Round Robin o FIFO, consiste en que cada petición del cliente se envía a un servidor, y la siguiente petición al siguiente servidor de la lista, hasta llegar al último tras lo cual se vuelve a enviar al primero.
Es la solución más sencilla y que menos recursos consume, aunque no es la más conveniente, ya que es posible que toda la carga pesada vaya a parar al mismo servidor mientras que el resto sólo reciban peticiones triviales.
Otro problema de este método, es que todos los servidores recibirán el mismo número de peticiones, independientemente de si su potencia de cálculo es la misma o no.

2.10.3.2 Round Robin (Ponderado)

Este algoritmo es igual que el anterior, pero añadiendo un “peso” a cada servidor. Este peso es un valor entero que indica la potencia de cálculo del servidor, de forma que la cola Round Robin se modificará para que aquellos servidores con mayor potencia de cálculo reciban peticiones más a menudo que el resto.
El problema de este método es que, si bien asegura que los servidores más capaces reciban más carga, también por probabilidad acabarán recibiendo más peticiones “pesadas”, con lo que a pesar de todo podrían llegar a sobrecargarse.

2.10.3.3 Servidor con menos conexiones activas

Este mecanismo de distribución consulta a los servidores para verificar en cada momento cuántas conexiones abiertas tiene cada uno con los clientes, y envía cada petición al servidor que menos conexiones tenga en ese momento.
El problema de este método ocurre cuando la potencia de los servidores no es la misma, no así cuando todos tienen más o menos las mismas características.

2.10.3.4 Servidor con menos conexiones activas (Ponderado)

Este algoritmo es igual al anterior, pero integrando la estrategia Round Robin Ponderada, que consiste en añadir unos pesos a los servidores que de alguna forma midan su capacidad de cálculo, para modificar la preferencia a la hora de escoger uno u otro según este peso.

2.10.3.5 Menos conectado basado en servicio

Este método dirige todas las peticiones a un mismo servidor, hasta que se sobrecarga (su número de conexiones activas es mayor que su peso) y entonces pasa a una estrategia de menos conexiones activas ponderada sobre el resto de servidores del cluster. Este método de planificación puede ser útil cuando ofrecemos varios servicios distintos y queremos especializar cada máquina en un servicio, pero siendo todas ellas capaces de reemplazar a las demás.

Conexiones persistentes

A todos los algoritmos de planificación anteriores, se les puede añadir que una vez que un cliente se ha conectado con un servidor, siempre se le dirija al mismo servidor. Este algoritmo hace uso de una tabla de asignaciones fijas.

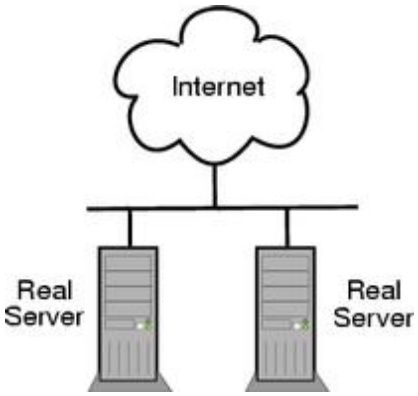
2.11 Tipos y topologías de los clusters de Alta Disponibilidad.

Las topologías de los clusters de Alta Disponibilidad se dividir en dos tipos:

- Estándar
  - Servicio simple con Alta Disponibilidad
  - Servicio con Balanceo de Carga y Alta Disponibilidad
- Avanzada
  - Servicio con Balanceo de carga, alta disponibilidad y alta capacidad.
  - Flujo de Alta Disponibilidad y Balanceo de Carga.

Servicio Simple con Alta Disponibilidad

Esta topología provee un servicio de alta disponibilidad con requerimientos mínimos de hardware, posee dos servidores reales monitoreándose el uno al otro, en el momento que el servidor activo cae, el servidor de backup asume la dirección virtual del master y el servicio es mantenido.
Es un caso muy sencillo en el que ni siquiera se utiliza balanceador de carga, tan sólo dos servidores que se monitorizan mutuamente.

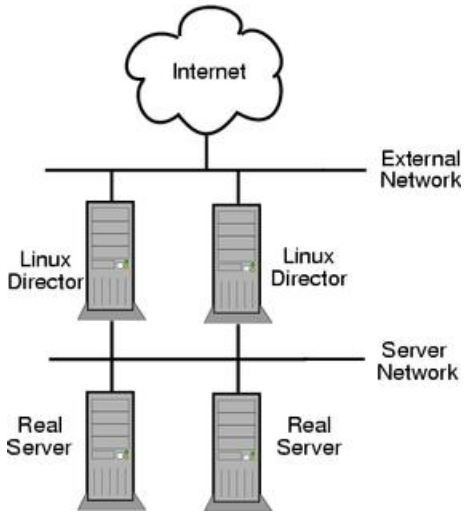


Servicio con Balanceo de Carga y Alta Disponibilidad

Esta topología provee un servicio de Alta Disponibilidad con Balanceo de Carga. Para esta topología se requieren al menos 4 nodos. Los servidores reales pueden ser agregados a la red para obtener capacidad adicional si se requiere.

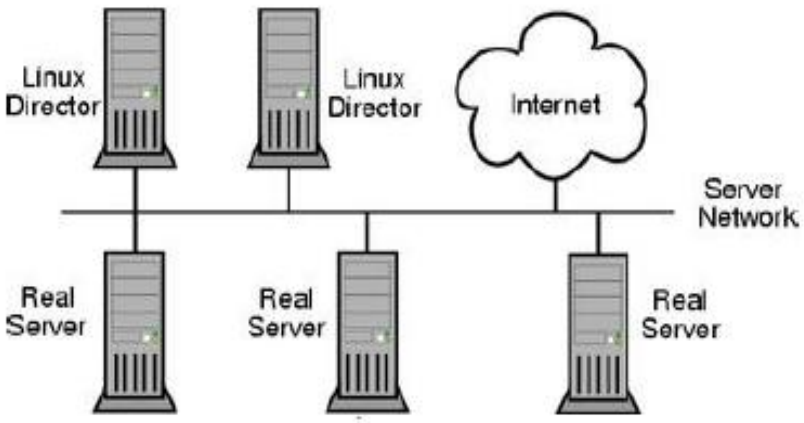
Esta topología es similar a la anterior, pero añadiendo al frente de todo un balanceador de carga, y separando así a los servidores reales de Internet. El balanceador controla el estado de los servidores y utiliza el método de direccionamiento VS- NAT.

Además se coloca un equipo de respaldo para el balanceador de carga, que se encargará de monitorizarlo y tomar su lugar con fakeover ante cualquier problema.



Servicio con Balanceo de carga, alta disponibilidad y alta capacidad

Similar a la anterior, pero los servidores y los balanceadores de carga se sitúan en la misma red con salida a Internet, para conseguir así un mayor ancho de banda de salida utilizando VS-Tun o VS-DR.



Flujo de Alta Disponibilidad y Balanceo de Carga

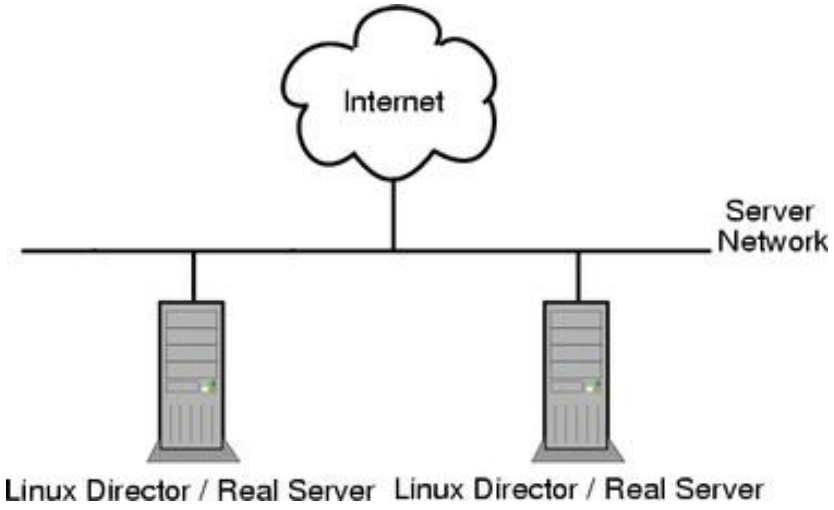
Esta topología combina la función de Director de Linux y Servidor Real in el mismo host, aquí la alta disponibilidad y balanceo de carga puede ser obtenida solo con dos nodos.

El director de linux activo acepta el tráfico para una dirección IP virtual. Los dos nodos se supervisan usando el Latido del corazón y en caso que el director de linux activo falla el otro nodo asume la dirección virtual y el servicio se mantiene.

El nodo director supervisa el estado de los servicios en ambos nodos pidiendo una página conocida periódicamente y verificando la contestación. Si un servicio falla en un servidor, entonces el servidor se saca del arreglo de servidores reales y se reinserta una vez que regresa en línea.

El camino del retorno para las conexiones está directamente dirigido hacia la red del área local. Cuando se usa la asignación de ruta directa en esta configuración, los paquetes del retorno no necesitan atravesar el balanceador de carga.

Los servidores pueden ejecutar una variedad de servicios incluso El Servidor de HTTP apache. Los servidores reales adicionales pueden agregarse a la red para obtener la capacidad que se requiere.



2.12 ¿Por qué construir un cluster de Alta Disponibilidad?

Entre las razones que podemos citar del por qué construir un cluster de Alta Disponibilidad tenemos dos estudios realizados independientemente en el año de 1995 por las compañías Oracle Corp y Datamation que indican que una empresa media pierde entre 80,000 y 350,000 dólares por hora de interrupción no planeada de sus servicios informáticos.

Otro ejemplo de la necesidad de la alta disponibilidad es que tras el atentado en el World Trade Center en 1993, 145 de las 350 empresas que allí se hospedaban (algo más del 40%) tuvieron que cerrar sus puertas tras este incidente por no disponer de una infraestructura informática redundante. [5]

2.13 Intereses Comerciales

El mercado de clusters de Alta Disponibilidad es un mercado con muchos clientes potenciales, siendo la mayoría de estos clientes empresas. Existen empresas que poseen ordenadores carísimos ejecutando sus operaciones fundamentales, estos ordenadores no deben caer nunca, ya que si esto ocurriera, el funcionamiento de la empresa, se vendría abajo, ocasionando grandes caos.

Debido a esta razón se desarrollan proyectos que intentan conseguir esta disponibilidad, pero no gracias a un soporte de hardware carísimos sino usando conceptos de clusters. Las empresas que requieren de alta disponibilidad suelen pagar a empresas que les ofrece alta disponibilidad aun cuando los programas sean de libre distribución porque necesitan garantía.

Las grandes diferencias entre el precio del hardware de las soluciones tradicionales y estas nuevas soluciones hacen que las empresas tengan un buen margen de beneficio, por lo que todas las empresas hacen sus propias soluciones o cogen las soluciones libres existentes dando un servicio de soporte.

Con la replicación de sistemas de archivos, bases de datos o servicios necesarios para que éstos estén disponibles, se logra mantener que la empresa este en funcionamiento todos y cada uno de los días lo que supone una mayor inversión en este tipo de clusters. [6]

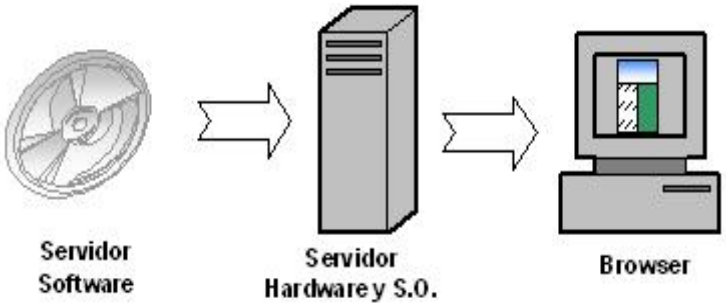
CAPITULO III: SERVIDOR WEB

3.1 Servidor Web

El servidor Web es un programa que corre sobre el servidor que escucha las peticiones HTTP que le llegan y las satisface. Dependiendo del tipo de la petición, el servidor Web buscará una página Web o bien ejecutará un programa en el servidor. De cualquier modo, siempre devolverá algún tipo de resultado HTML al cliente o navegador que realizó la petición.

Un servidor Web esta formado por:

- Programa Servidor (Software Apache, Netscape)
- Computador Servidor (Hardware y Sistema Operativo)
- Contenido Web



El mayor problema aislado que afecta al rendimiento de los servidores Web es la RAM. Un servidor Web nunca jamás debería tener que hacer intercambio (swapping). El intercambio incrementa la latencia de cada petición más allá del punto que los usuarios consideran como "suficientemente rápido".

El resto es vano: conseguir una CPU suficientemente rápida, una tarjeta de red suficientemente rápida, y discos suficientemente rápidos, donde "suficientemente rápido" es algo que necesita determinarse mediante experimentación.

3.1.1 Apache

Apache es el software por excelencia utilizado para dar servicio de Web en el mundo Linux (existen versiones para otras plataformas como Windows, etc.), porque es un software robusto y bien probado.

El objetivo del servidor Web Apache está diseñado, en primer lugar para funcionar correctamente y después rápido. Aún así, su rendimiento es bastante satisfactorio.

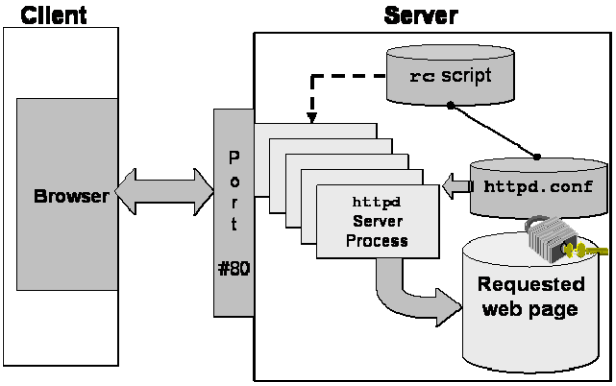
3.1.1.1 Arrancando Apache en Unix

El proceso httpd corre como demonio en Linux, el cual se ejecuta continuamente en background para manipular peticiones. Apache puede ser invocado por el demonio Internet inetd cada vez que se hace una conexión al servicio HTTP usando la directiva ServerType.

El puerto por defecto utilizado por Apache, es el 80 (o cualquier otro por debajo de 1024), además es necesario tener privilegios de root para iniciar Apache.

Una vez que el servidor ha arrancado y completado algunas actividades preeliminares como la apertura de sus ficheros de log, lanzará muchos procesos hijo que hacen el trabajo de escuchar y responder las peticiones de los clientes.

El proceso principal de Apache es httpd, el cual continúa corriendo como usuario root, pero los procesos hijo se ejecutan con un usuario con menos privilegios.



Lo primero que hace httpd cuando es invocado es localizar y leer el fichero de configuración httpd.conf. La localización de este fichero es determinada en el

momento de la compilación, pero es posible especificar su situación en el momento de ejecución, usando la opción -f en la línea de comandos de la forma.
/usr/local/apache/bin/httpd -f /etc/httpd/conf/httpd.conf

3.1.1.2 Errores que se producen durante El Arranque

Cuando Apache sufre un problema grave durante el arranque, éste se indicará en la consola, o en el ErrorLog antes de salir y terminar. Entre los errores más comunes tenemos "Unable to bind to Port (No se pudo hacer bind al puerto...)". Este mensaje se debe normalmente por:

- Intentar arrancar el servidor en un puerto privilegiado cuando no se está autenticado como usuario root.
- Intentar arrancar el servidor cuando hay otra instancia de Apache o cualquier otro servidor Web ya enganchado al mismo puerto.

3.1.1.3 Ficheros de Configuración

Principales ficheros de configuración

Módulos asociados	**Directivas relevantes**
mod_mime	<IfDefine> Incluye TypesConfig

Apache se configura poniendo directivas en los ficheros de configuración en texto simple. El fichero principal de configuración es httpd.conf. La ubicación de este fichero se define cuando se compila Apache. Algunos servidores tienen también ficheros srm.conf y access.conf por razones históricas. Además de eso, se pueden adicionar otros ficheros de configuración usando la directiva Include.

Apache tiene la funcionalidad en la que si cualquiera de los ficheros de configuración es un directorio, Apache entrará en ese directorio y analizará cualquier fichero (y subdirectorio) que se encuentre allí, tratándolos como ficheros de configuración.

Módulos

Módulos asociados	Directivas relevantes
mod_so	ClearModuleList <IfModule> LoadModule

Apache es un servidor modular, es decir que el servidor básico incluye únicamente las funcionalidades más básicas. Otras funcionalidades se encuentran disponibles a través de módulos que pueden ser cargados por Apache. Si el servidor se compila para usar carga dinámica de módulos, entonces los módulos pueden ser compilados por separado, e incluidos en cualquier momento usando la directiva LoadModule.

Rango de acción de las Directivas

Módulos asociados	Directivas relevantes
	<Directory> <DirectoryMatch> <Files> <FilesMatch> <Location> <LocationMatch> <VirtualHost>

Las directivas que se coloquen en los ficheros principales de configuración se aplicarán a todo el servidor, pero si lo que se desea es cambiar únicamente la configuración de una parte del servidor, se puede cambiar el rango de acción de las directivas poniéndolas dentro de las secciones <Directory>, <DirectoryMatch>, <Files>, <FilesMatch>, <Location>, y <LocationMatch> . Estas secciones limitan el dominio de aplicación de las directivas dentro de ellas, a locales particulares dentro del sistema de ficheros o URL's particulares.

Otra de las características de Apache es servir a diferentes sitios Web al mismo tiempo. Esto se llama Hospedaje Virtual. El dominio de aplicación de las directivas también puede ser delimitado poniéndolas dentro de <VirtualHost>, de manera que solo tendrán efecto para pedidos de un sitio Web en particular.

Ficheros .htaccess

Módulos asociados	Directivas relevantes
	AccessFileName
	AllowOverride

Apache permite una administración descentralizada de la configuración, a través de ficheros colocados dentro del árbol de páginas Web. Los ficheros especiales se llaman normalmente .htaccess, pero se puede especificar cualquier otro nombre en la directiva AccessFileName. Las directivas que se pongan dentro de los ficheros .htaccess se aplicarán únicamente al directorio donde esté el fichero, y a todos sus subdirectorios. Los ficheros .htaccess siguen las mismas reglas de sintaxis que los ficheros principales de configuración.

Configuración global del servidor

Identificación del servidor

Módulos asociados	Directivas relevantes
	ServerName
	ServerAdmin
	ServerSignature
	ServerTokens
	UseCanonicalName

Las directivas ServerAdmin y ServerTokens controlan la información del servidor que debe ser presentada en documentos generados por el servidor, como por ejemplo mensajes de error.
La directiva ServerTokens fija el contenido del campo de encabezado en la respuesta del servidor HTTP.

El servidor usa las directivas ServerName y UseCanonicalName para determinar como construir URLs con referencias a ellas mismas.

Localización de ficheros

Módulos asociados	Directivas relevantes
	CoreDumpDirectory
	DocumentRoot
	ErrorLog
	Lockfile
	PidFile
	ScoreBoardFile
	ServerRoot

Estas directivas controlan la localización de los ficheros que Apache necesita para su correcto funcionamiento. Cuando el camino usado no comienza con un slash "/", los ficheros son localizados relativamente al valor de ServerRoot.

Virtual Host

El término Virtual Host se refiere a que esté corriendo más de un sitio Web en una única máquina. Virtual hosts puede estar basado en IP, lo que significa, que se tiene diferentes direcciones IP para cada sitio Web, o puede estar basado en nombre, es decir, se tiene varios nombres corriendo para cada dirección IP.
Por lo tanto, ellos están corriendo en el mismo servidor físico, lo cual es transparente para el usuario final.

Para configurar un Virtual Host debemos agregar al final del archivo de configuración httpd.conf

```
<VirtualHost *:80>
        ServerName          Nombre_servidor_virtual
        DocumentRoot        /var/www/Nombre_servidor_virtual
        ServerAdmin         admin@Nombre_servidor_virtual
</VirtualHost>
```

Nota: * Utiliza todas las direcciones IP
*:80 Utiliza todas las direcciones IP, pero solo por el puerto 80.

Luego creamos el directorio especificado en el DocumentRoot:

cd /var/www/

mkdir Nombre_sevidor_virtual

Dentro de este directorio colocamos las páginas del Virtual Host

En el archivo httpd.conf agregamos la siguiente línea:

NameVirtualHost *:80

Y por último reiniciamos el servicio.

Contenido dinámico con CGI

Módulos asociados	**Directivas relevantes**
mod_alias	AddHandler
Mod_cgi	Options
	ScriptAlias

CGI (Common Gateway Interface) es una manera como un servidor Web interactúa con programas externos generadores de contenido, los mismos que son conocidos como CGI o CGI scripts. Este es el modo más común y simple de crear contenido dinámico en tu sitio Web.

Para conseguir que los programas CGI funcionen correctamente, se necesita que Apache esté configurado para permitir la ejecución CGI.

La directiva ScriptAlias indica que Apache posee un directorio concreto para programas CGI. Apache asumirá que cada fichero de este directorio es un programa CGI, e intentará ejecutarlo, cuando este recurso particular sea solicitado por un cliente.

La directiva ScriptAlias se muestra así:

ScriptAlias /cgi-bin/ /var/www/cgi-bin/

ScriptAlias es utilizado normalmente para directorios que están fuera del directorio DocumentRoot. ScriptAlias tiene el significado adicional de que todo bajo el prefijo URL se considera un programa CGI.

3.1.1.4 Ficheros de Bitácora (Log)

Fichero pid.- Al arrancar Apache almacena el número del proceso padre del httpd en el fichero logs/httpd.pid. Este nombre de fichero se puede cambiar con la directiva PidFile. El número del proceso es para el uso del administrador, cuando quiera terminar o reiniciar el demonio. Si el proceso muere anormalmente, entonces será necesario matar los procesos hijos.

Bitácora de errores.- El servidor registrará los mensajes de error en un fichero de bitácora, que será por defecto logs/error_log. El nombre del fichero se puede alterar usando la directiva ErrorLog, se pueden usar diferentes bitácoras de error para diferentes anfitriones virtuales.

Bitácora de transferencia.- El servidor normalmente registrará cada pedido en un fichero de transferencia que, por defecto, será logs/access_log. El nombre del fichero también se puede alterar usando la directiva CustomLog; se pueden usar diferentes ficheros de transferencia para diferentes anfitriones virtuales.

3.1.1.5 Parando y rearrancando Apache

Puede existir varios ejecutables httpd corriendo en el sistema, pero no debe enviar señales a ninguno excepto al padre, cuyo PID está en el Archivo PID. Hay tres señales que puede enviar al padre: TERM, HUP, y USR1, que son:

Señal TERM (terminar ahora)

Al enviar la señal TERM al proceso padre, éste mata de inmediato a todos los procesos hijo. Luego el proceso padre en sí termina. Cualquier requerimiento en progreso es finalizado, y ningún nuevo requerimiento es servido.

Señal HUP (reiniciar ahora)

Enviando la señal HUP al proceso padre provoca que todos los hijos sean terminados como con la señal TERM pero el padre no termina. El cual lee nuevamente sus archivos de configuración y vuelve a abrir los archivos log. Luego arranca una nueva serie de procesos hijo y continúa atendiendo peticiones.
Si los archivos de configuración contienen errores, el proceso padre no rearrancará cuando se reinicie el servidor.

Señal USR1 (rearranque agraciado)

La señal USR1 causa que el proceso padre notifique a los procesos hijos que terminen el requerimiento actual. El padre lee nuevamente sus archivos de configuración y vuelve a abrir los archivos log. Como cada hijo muere, el padre los reemplaza con hijos con nueva configuración, los cuales comienzan a servir los nuevos requerimientos inmediatamente.

3.2 Servidor Mail

3.2.1 Sendmail

Existe una gran variedad de programas de correo electrónico que proveen al usuario de una aplicación para la creación y envío de mail. Estos programas son llamados Agentes de Usuario o MUA (Mail User Agent), cuyo propósito es el aislar al usuario de los Agentes de Transporte o MTA (Mail Transport Agent), que son los encargados de transferir los mails a su correcto destino.

Sendmail es el agente de transporte de correo más común de Internet (en los sistemas UNIX). Las misiones básicas de sendmail son las siguientes:

- Recoge los mails provenientes de un Mail User Agent (MUA), o provenientes de un Mail Transport Agent (MTA) como puede ser el propio sendmail.

- Elección de la estrategia de reparto de los mails, basándose en la información de la dirección del destinatario contenida en la cabecera.

- Si el mail es local, enviará el mail al programa de reparto local de mails.

- Si el mail no es local, sendmail utiliza el DNS del sistema para determinar el host al que debe ser enviado el mail. Para transferir el mensaje, inicia una sesión SMTP con el MTA de dicho host.

- Si no es posible mandar el mail a su destino, sendmail almacenará los mails en una cola de correo, y volverá a intentar el envío del mail un tiempo después. Si el mail no puede ser enviado tras un tiempo razonable, el mail será devuelto a su autor con un mensaje de error. Sendmail debe garantizar que cada mensaje llegue correctamente a su destino, o si hay error este debe ser notificado.

- Reformatear el mail antes de enviarlo a la siguiente máquina, según unas reglas de reescritura. Según el tipo de conexión que posea determinada máquina, o según el agente de transporte al que vaya dirigido el mail, se necesitará cambiar los formatos de las direcciones del remitente y del destinatario, algunas líneas de la cabecera del mail, o incluso puede que necesitemos añadir alguna línea a la cabecera. Sendmail debe realizar todas estas tareas para conseguir la máxima compatibilidad entre usuarios distintos.

- Otra función muy importante de sendmail es permitir el uso de "alias" entre los usuarios del sistema; lo que nos permitirá crear y mantener listas de correo entre grupos.

- Ejecución como agente de usuario (MUA). Aunque no posee interfaz de usuario, sendmail también permite el envío directo de mails a través de su ejecutable.

Protocolos

SMTP

SMTP (Simple Mail Tranfer Protocol), es un protocolo básico para transferir mensajes de correo electrónico hosts IP, estableciendo conexiones al puerto 25 de TCP.

El intercambio de mails se basa en tres protocolos:

- Un standard para el intercambio de mails entre 2 computadoras, conocido como SMTP.
- Un standard que define el formato de los mensajes de mail. RFC 822 describe la sintaxis de los headers y RFC 1049 describe como encapsular documentos que no sean texto en el body del mail.

- Un estándar para el rounting de mail usando DNS, descrito en el RFC 974. El nombre oficial de este protocolo es DNS-MX.

Modelo de comunicación

- La entidad SMTP que envía los mensajes de correo establece una conexión con el SMTP receptor.

- El receptor puede ser el destino final o un gateway.

- El que envía genera una serie de comandos que serán respondidos por el receptor.

POP3

El Post Office Protocol una estable conexión TCP usando el port 110 en el servidor. Permitiendo que el cliente descargue mensajes de mail de su casilla en servidor.
Solo soporta copiados y borrados de mensajes en el servidor. Implementando mecanismos de autenticación entre el cliente y el servidor.

IMAP

El Internet Message Access Protocol, al igual que POP, pero una versión mejorada, permitiendo descargar mensajes por pedido. Restablece sincronismo entre la información del cliente y la del servidor. Mantiene siempre una copia de los mensajes en el servidor. También nos permite crear carpetas.

Mensajes de Mail

Header (cabecera)

Esta es una sección informativa que contienen todos los mail y que contiene datos relacionados a su envío, tales como el nombre y dirección electrónica del creador del mensaje, la lista de destinatarios, la fecha de envío, los servidores intermedios por donde el mensaje ha pasado, etc.

Body

Contiene el texto del mail.

Envelope (sobre)

Contiene información usada para enrutar los mensajes, tal como los destinatarios inmediatos. Esta información normalmente tiene coincidencias con algunos componentes del header.

Attachment

Los archivos que no se componen de texto ASCII pueden ser enviados si primero se codifican como texto ASCII y se añaden ordenadamente a un mensaje normal. Estos añadidos al mensaje se denominan attachments.

Casilla del usuario

Los usuarios de correo electrónico no están conectados a la red todo el tiempo. Debido a esto, los mensajes que están dirigidos a ellos normalmente se almacenan en un área temporal denominada "casilla de correo" a la espera de que el usuario la extraiga cuando esté listo.

Relay

Corresponde a la facultad del MTA de reenviar los mensajes provenientes de un computador hacia otro computador. Por ejemplo, cuando un usuario de la red local le proporciona un mensaje que en realidad está dirigido hacia un usuario en Internet.

3.2.2 Web Mail

WebMail es una tecnología que permite acceder a una cuenta de Correo Electrónico a través de un navegador de Internet. De esta forma se puede acceder a nuestra casilla de correo desde cualquier parte del mundo.
WebMail utiliza la tecnología IMAP, que permite leer y organizar los mensajes en carpetas sin descargar la información en la PC, es decir nos permite leer nuestro email desde cualquier computadora conectada al Internet sin tener que dejar nuestros mensajes privados de email en otra computadora.

Tiene prácticamente los mismos elementos que un programa cliente de correo, el cuál consta en la parte derecha de una pantalla en la que se puede ver la bandeja de entrada con los correos que han llegado a la cuenta, además podemos observar el listado de las diferentes carpetas. En la parte superior existen unas viñetas para redactar un nuevo correo, mostrar encabezados, salir del webmail, organizar por fecha de recibido, eliminar todos los mensajes, marcar los mails como leídos o mover a otros fólderes.

Esta forma de usar el correo surgió debido a la limitación que imponen los programas de correo de tener que configurarlos en cada ordenador donde se utilicen. Para ingresar al Webmail se debe ingresar el usuario y la contraseña.

3.2.2.1 Squirrelmail

Squirrelmail es un programa gestor de correo IMAP (Internet Message Access Protocol), que es un protocolo de red de acceso a mensajes electrónicos almacenados en un servidor.

Luego de configurar la cuenta IMAP, se puede especificar las carpetas que se desean mostrar y las que se desean ocultar, esta característica lo hace diferente del protocolo POP.

3.2.2.2 Características de Squirrelmail

Entre las características más destacadas de Squirrelmail tenemos:

- Gestión de Carpetas.
- Internacionalización.
- Libro de direcciones personal y acceso a otros servicios de LDAP.
- Búsquedas de direcciones.
- Servicio de búsqueda en emails.
- Interfaz de usuario sencilla y potente.
- Gestión de attachments.
- Configuración de las vistas de mensajes: número de mensajes visibles en pantalla, campos visibles y orden.
- Comprobación de correos entrantes cada cierto intervalo de tiempo.

- Está escrito en PHP4 y es GPL, por lo que se puede ampliar, modificar fácilmente y es gratis.
- Arquitectura de plug-ins.
- Múltiples temas.
- No necesita ninguna base de datos para funcionar.

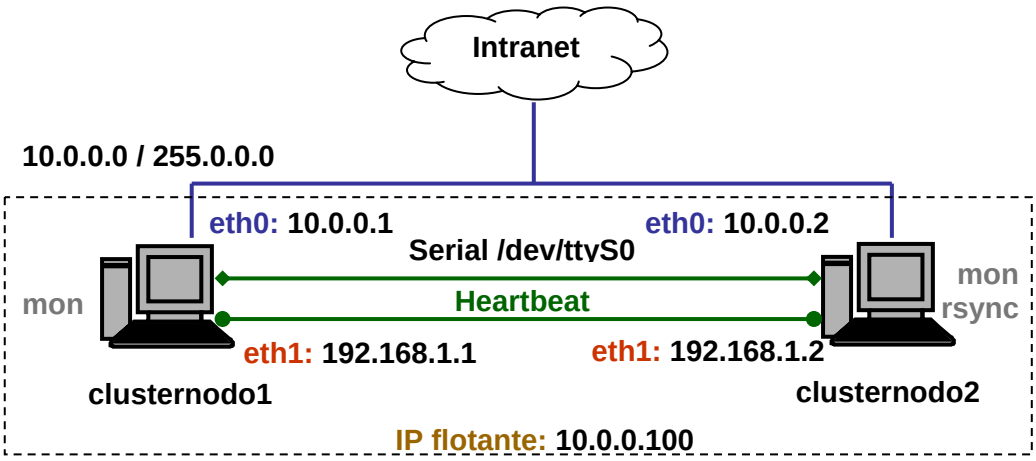
Debido a que Squirrelmail es un software fácilmente modificable, se pueden añadir algunos plug-ins para incluir nuevas funciones, por ejemplo:

- Auto completado de direcciones de correo cuando se escribe un email.
- Envío de páginas HTML comprimidas.
- Filtros de mensajes según direcciones de correo o subject.
- Filtrado de spamming.
- Corrección de correos en cualquier idioma.
- Traducción de correos a diferentes lenguajes.

CAPITULO IV: IMPLEMENTACIÓN DE UN CLUSTER HA

La práctica que hemos realizado, consiste en la implementación de un Cluster de Alta Disponibilidad Activo/Pasivo para brindar los servicios de Web (Apache) y Mail (Sendmail), para el dominio *cluster.com.ec.* Para esto, se ha utilizado la topología de *Servicio Simple con Alta Disponibilidad*, que necesita dos nodos para conformar el cluster, la primera máquina será el nodo maestro, mientras que la segunda será el nodo de respaldo. El trabajo del nodo de respaldo es hacerse cargo de los servicios de httpd, sendmail y dns del nodo maestro si dicha máquina ha fallado. Para esto, se ha configurado de igual manera, todos los servicios en ambos nodos.

Además, se ha configurado, una serie de herramientas tales como: Sincronizador de Archivos *(Rsync)*, Monitoriador de Hardware *(Heartbeat)* y Monitoriador de Servicios *(Mon)*, para obtener alta disponibilidad.



Cortéz Guerrero, Stanley
Galarza Rodas, Mónica

4.1 Hardware Implantado

A continuación se indica las características del Hardware requeridos para la implementación del cluster:

4.1.1 Nodos

Nodo1: Llamado Clusternodo1

Procesador:	Intel(R) Pentium(R) 4 CPU 2.5 GHz
Memoria RAM:	512 Mb
Disco Duro:	40 Gb
Tarjeta de Red (2):	10/100Mbps Fast Ethernet Card PCI Adapter, con chip Davicom Semiconductor.

Nodo2: Llamado Clusternodo2

Procesador:	Intel(R) Pentium(R) 4 CPU 1.7 GHz
Memoria RAM:	256 Mb
Disco Duro:	40 Gb
Tarjeta de Red (2):	10/100Mbps Fast Ethernet Card PCI Adapter, con chip Davicom Semiconductor.

4.1.2 Clientes

Cliente1: Llamado Cliente

Procesador:	AMD-K6(tm) 3D processor CPU 500 Mhz
Memoria RAM:	320 Mb
Disco Duro:	60 Gb
Tarjeta de Red (1):	10/100Mbps Fast Ethernet Card PCI Adapter, con chip Davicom Semiconductor.

4.1.3 Red

Hub

16-Port Mini-Hub ESH-717

Marca ENCORE

Cables

Cable UTP, categoría 5, 6

1 cable punto a punto.

3 cables derechos.

1 cable serial.

4.2 Configuración de la Red

Para migrar los procesos se necesita un soporte TCP/IP de transmisión de datos, para lo cual configuramos la red local para obtener una comunicación ininterrumpida entre los nodos.

Ambas máquinas que forman el cluster poseen dos tarjetas de red cada una y al menos un puerto serie. La primera tarjeta de red de cada máquina (eth0) tendrá la dirección IP de la red local y la segunda tarjeta (eth1) tendrá la dirección IP de la red privada que se encargará del latido.

La interfaz eth0 será en cada nodo de la siguiente manera:

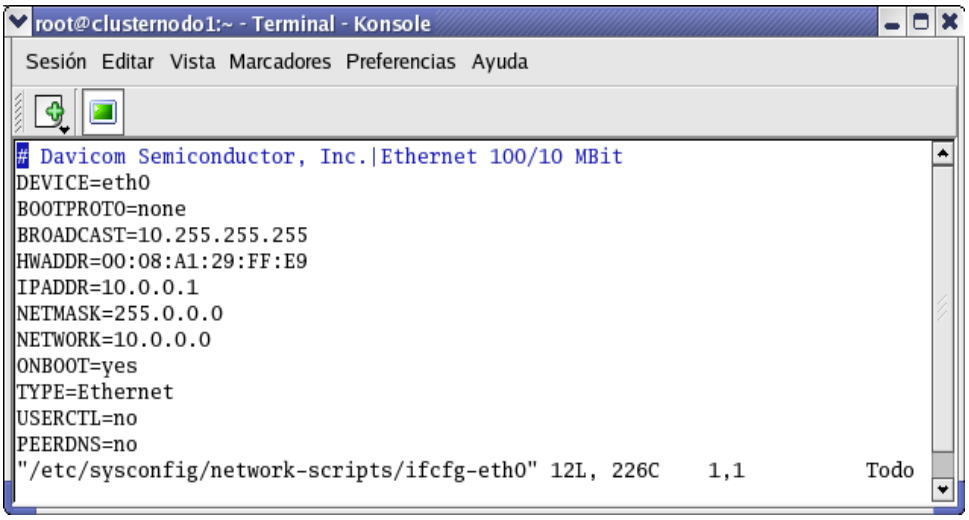
Red	10.0.0.0
Máscara	255.0.0.0
Broadcast	10.255.255.255

clusternodo1 con dirección IP 10.0.0.1

clusternodo2 con dirección IP 10.0.0.2

Esta configuración se realiza en el siguiente archivo en cada uno de los nodos:

/etc/sysconfig/network-scripts/ifcfg-eth0: (p. ej clusternodo1)



Debemos reservar una dirección IP para uso flotante, que será la IP, que responderá a los servicios, y será *10.0.0.100*. De momento no se le asigna a ninguna máquina.

La interfaz eth1 de ambos nodos, tomará direcciones IP con un rango fuera de uso:

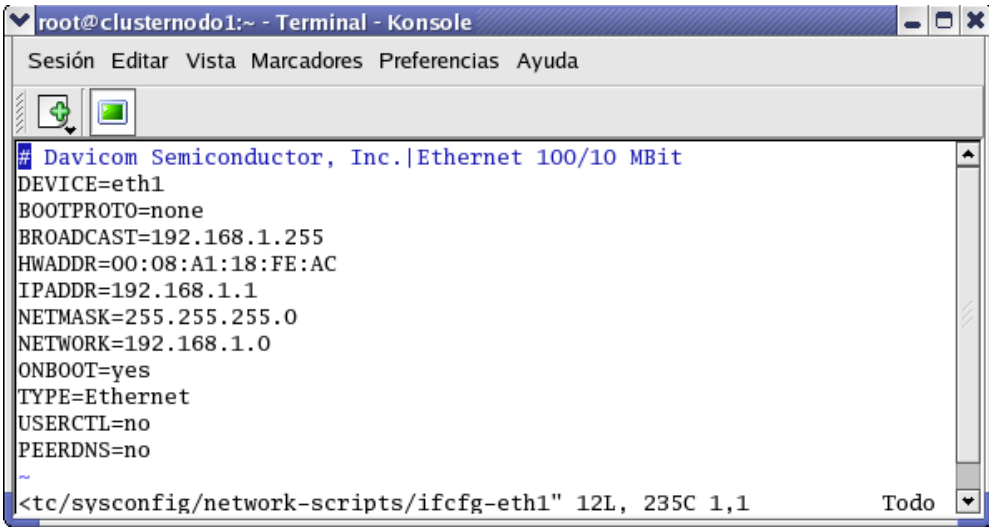
Red	192.168.1.0
Máscara	255.255.255.0
Broadcast	192.168.1.255

clusternodo1 con dirección IP 192.168.1.1

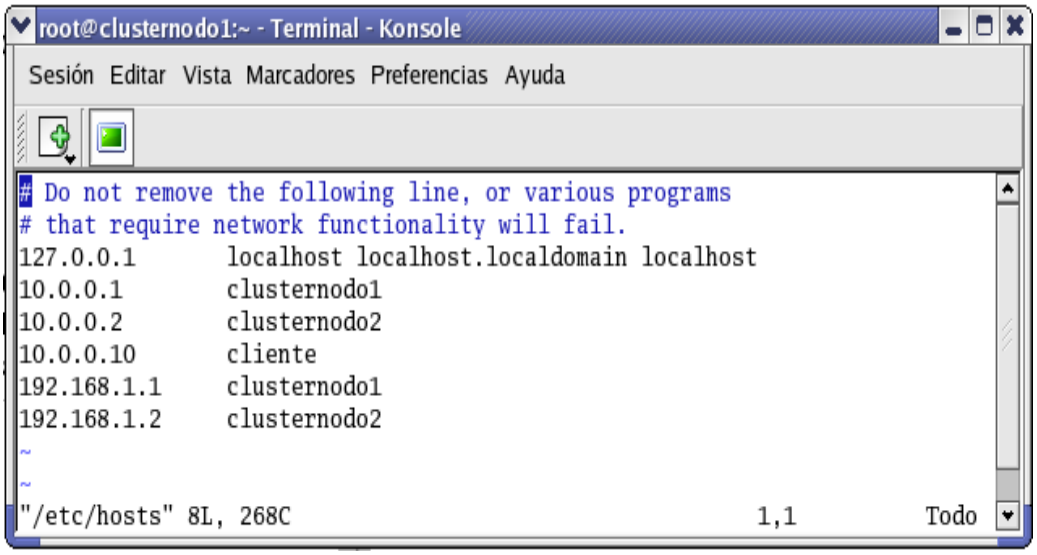
clusternodo2 con dirección IP 192.168.1.2

Esta configuración se realiza en el siguiente archivo en cada uno de los nodos:

/etc/sysconfig/network-scripts/ifcfg-eth1: (p.ej clusternodo1)



Ahora agregamos en el archivo */etc/hosts,* las direcciones IP con los respectivos nombres de todos los ordenadores de la red local para poder identificarlos. Este archivo se muestra de la siguiente manera:



Después de haber terminado la configuración en estos archivos, reiniciamos el servicio de red con la instrucción *service network restart.*

Para verificar que la red está funcionando correctamente podemos realizar la siguiente prueba:
ping 10.0.0.2 desde clusternodo1 (10.0.0.1) y si se obtiene respuesta del clusternodo2 (10.0.0.2), esto nos indica que existe conexión entre las máquinas.

4.2 Configuración del Software

Es muy importante instalar las mismas versiones del software en cada uno de los nodos para evitar problemas de incompatibilidad y para que el trabajo del cluster sea óptimo, consiguiendo de esta forma mejorar el rendimiento.

4.2.1 Distribución de Linux

La distribución del sistema Operativo que se ha escogido para la implementación del cluster es Fedora Core I, el cual ha sido instalado las mismas versiones en los dos equipos que forman el cluster, para evitar inconvenientes de incompatibilidad y para conseguir resultados más fiables a la hora de realizar las pruebas.

Versión del Kernel

El kernel de Linux, es decir el núcleo del sistemas es la parte más importante del sistema operativo, ya que se encarga de llevar el control de la memoria, los procesos y el hardware del sistema. Existen distintas versiones del kernel debido a las nuevas mejoras y nuevas prestaciones que se van agregando periódicamente, por lo tanto la versión del núcleo de Linux que se ha instalado es 2.4.22-1.2115.nplt

4.2.2 Configuración de servidores.

La configuración de los servidores que se indican a continuación, debe ser realizada en ambos nodos del cluster, ya que si el nodo maestro falla, el nodo respaldo debe brindar los mismos servicios, de manera automática, por este motivo se deben realizar las siguientes configuraciones en clusternodo1 y en clusternodo2.

4.2.2.1 Servidor DNS

Configuración

Para configurar este servidor se debe realizar lo siguiente:

/etc/named.conf

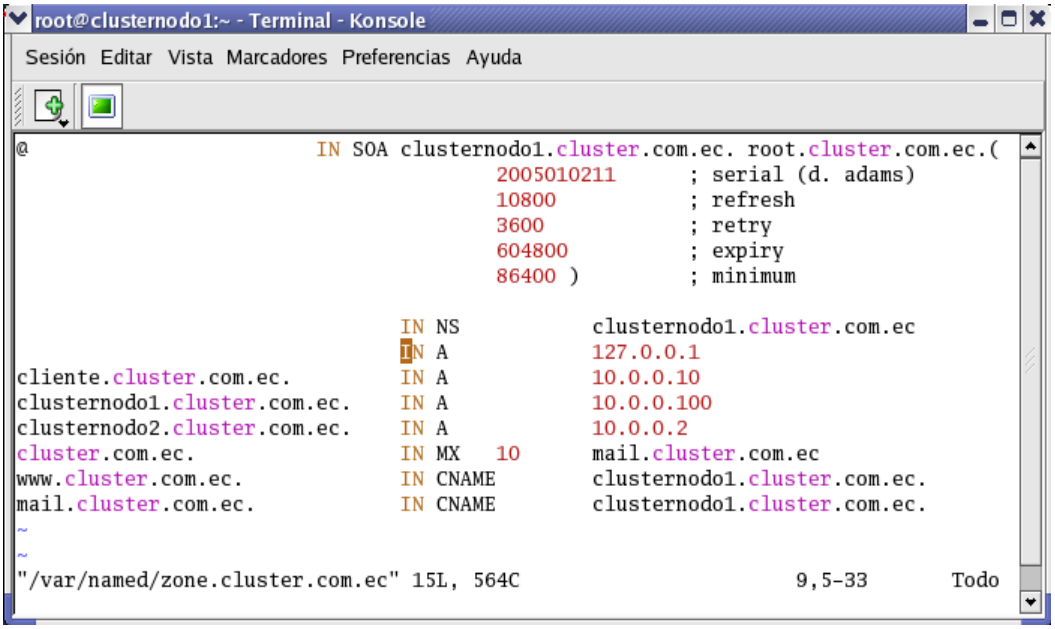
Definimos la *zona "cluster.com.ec"*, en el archivo */etc/named.conf, de la siguiente manera:*

```
zone "cluster.com.ec" IN {
        type master;
        file "zone.cluster.com.ec";
};
zone "0.0.0.10.in-addr.arpa" IN {
        type master;
        file "zone.0.0.10.in-addr.arpa";
};
```

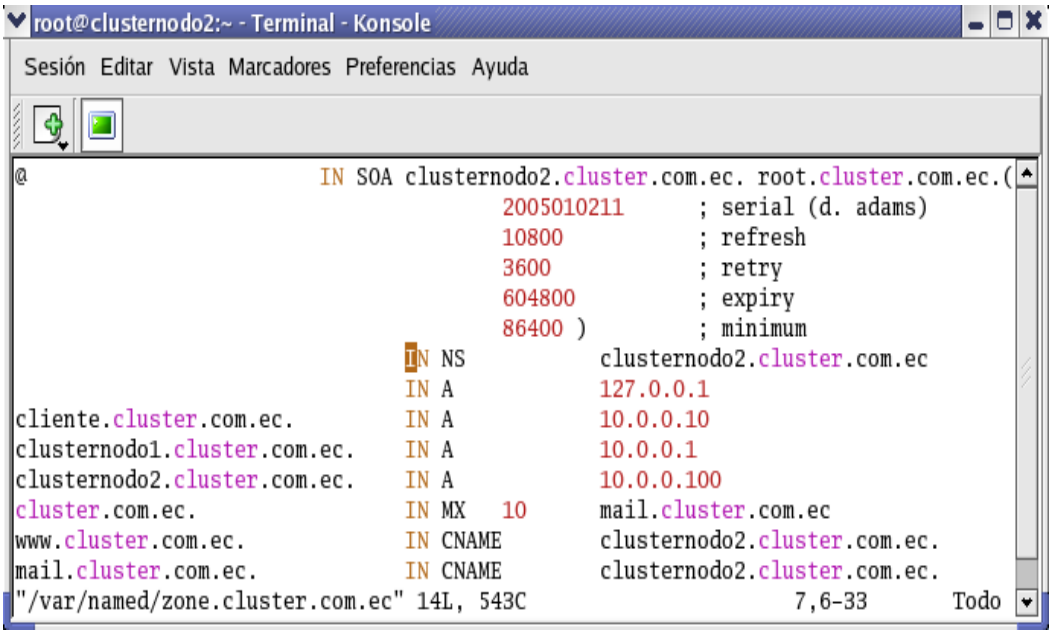
/var/named/zone.cluster.com.ec

Creamos el archivo *zone.cluster.com.ec* en la dirección */var/named*, que contendrá el registro del servidor DNS, que es editado para este cluster de la siguiente manera:

En clusternodo1:



En clusternodo2:



En el archivo hemos agregado los siguientes registros:

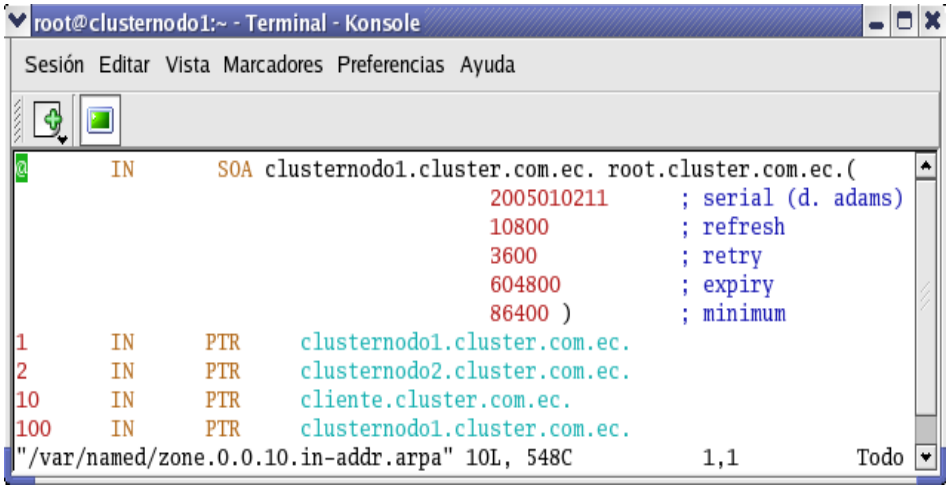
NS	Indica cual es el servidor de la zona.
MX	Establece cual es el servidor de mail del dominio cluster.com.ec.
A	Indica las direcciones IP de los hosts correspondientes a la zona, que será administrada por el DNS, incluyendo la dirección IP de la misma máquina.
CNAME	Pone un alias a un host.

Como se mencionó anteriormente, la IP 10.0.0.100, que responderá a los servicios, en este caso el servicio de resolución de nombres, deberá ser configurado (el registro A) en éste archivo, como se muestran en las figuras anteriores.

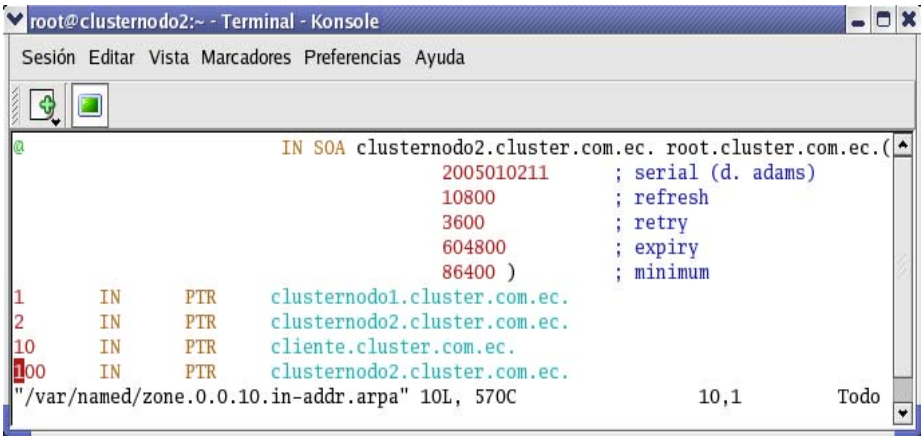
/var/named/zone.0.0.10.in-addr.arpa

Se crea el archivo zone.0.0.10.in-addr.arpa para la resolución inversa, es decir para que el DNS pueda permita resolver nombres a partir de direcciones IP.

En clusternodo1:



En clusternodo2:



/etc/resolv.conf

Se agrega la siguiente línea en el archivo */etc/resolv.conf,* de todos los host de la intranet.

[root@clusternodo1 root]#vi /etc/resolv.conf

```
nameserver 10.0.0.100
```

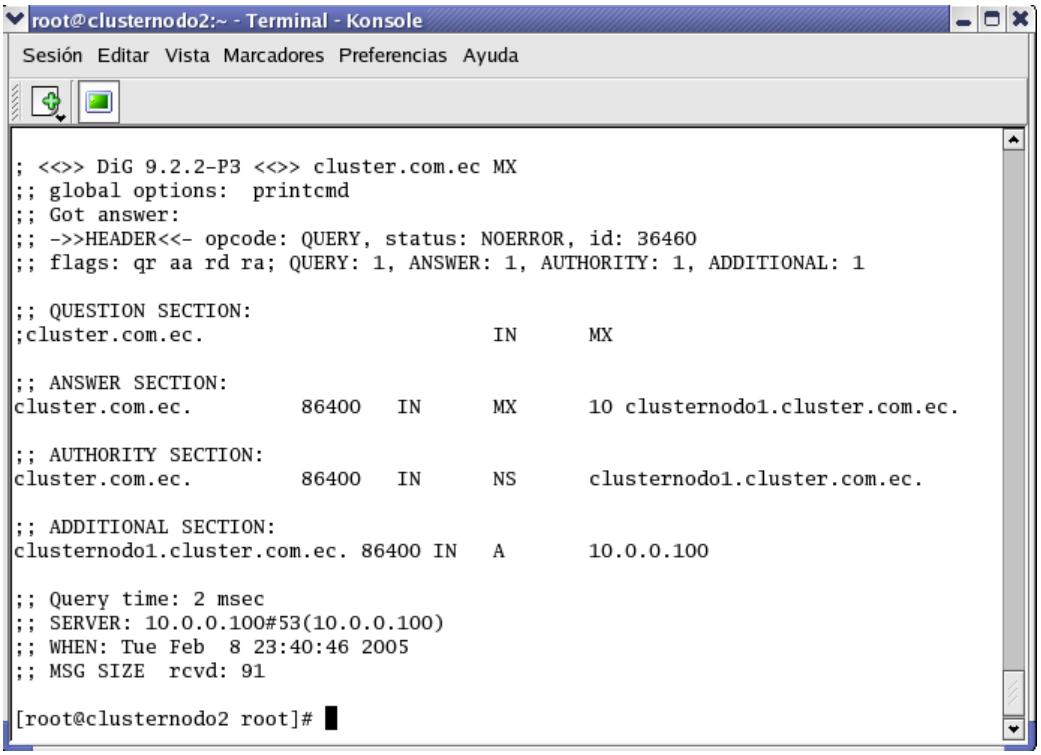
/etc/sysconfig/named

Ya que Fedora Core I, configura por defecto en el archivo */etc/sysconfig/named,* la dirección donde se encuentran los archivos de configuración, indicando que éstos están en *ROOTDIR=/var/name/chroot.* Colocamos esta línea con comentario para, que de esta manera se entienda que los archivos de configuración se encuentran en */var/named.*

Luego de terminar la configuración del Servidor DNS, procedemos a reiniciar el servicio con la siguiente instrucción:
[root@clusternodo1 root]#service named restart

Para comprobar que la configuración de DNS está correcta, colocamos en la línea de comandos, la siguiente instrucción:
[root@clusternodo1 root]#dig cluster.com.ec MX

Esta instrucción nos dará como respuesta, cuál es el servidor de mail del dominio cluster.com.ec, como se muestra a continuación:



4.2.2.2 Servidor Web

Para la configuración del servidor Web, utilizamos Apache, que es difundido en Internet de manera gratuita.

Configuración

/etc/httpd/conf/httpd.conf

En el archivo de configuración del servidor web */etc/httpd/conf/httpd.conf*, ubicamos y editamos las siguientes líneas:

DocumentRoot, en la cual especificamos cual es el directorio, en donde se alojan las páginas web del sitio *www.cluster.com.ec.*

DocumentRoot "/var/www/html"

ServerName, proporciona el nombre y puerto, que el servidor usa para identificarse.

ServerName 10.0.0.100:80

ScriptAlias, controla que directorios contienen los scripts.

ScriptAlias /cgi-bin/ "/var/www/cgi-bin/"

Por último reiniciamos el servicio utilizando el siguiente comando:

[root@clusternodo1 root]#service httpd restart

Para comprobar que la configuración se realizó correctamente, colocamos las páginas del sitio www.cluster.com.ec en la dirección especificada en el DocumentRoot.
En el browser escribimos, *http://www.cluster.com.ec,* y si todo está bien, veremos nuestra página principal.

4.2.2.3 Servidor de Mail

El servidor de Mail que se ha configurado, es Sendmail, el cual es un poderoso servidor de correo electrónico ampliamente utilizado.

Instalación

Confirmamos la instalación de Sendmail, utilizando la siguiente línea de comando:

[root@clusternodo1 root]#rpm -q sendmail sendmail-cf imap

Esto nos devolverá las versiones de sendmail, sendmail-cf e imap que se tienen instaladas:

sendmail-8.12.10-1.1.1

sendmail-cf-8.12.10-1.1.1

impap-2002d-3

Configuración

/etc/mail/local-host-names

Se ha editado el fichero */etc/mail/local-host-names* con la lista de todos y cada uno de los alias que tiene el servidor, el cual se está configurando (clusternodo1, clusternodo2), así como el dominio, del cual se recibirá correo.

[root@clusternodo1 mail]#vi local-host-names

```
# local-host-names - include all aliases for your machine here.
cluster.com.ec
clusternodo1.cluster.com.ec
```

/etc/mail/sendmail.mc

Modificamos el archivo */etc/mail/sendmail.mc*, de la siguiente manera:

Comentamos la siguiente línea con dnl, para poder enviar mails desde las máquinas de la red local.

dnl DAEMON_OPTIONS(`Port=smtp,Addr=127.0.0.1, Name=MTA')dnldnl

Como deseamos rechazar correo proveniente de dominios no resueltos, es necesario mantener comentada la siguiente línea:

dnl FEATURE(`accept_unresolvable_domains')dnl

Es necesario establecer que dominio, corresponderá a la máscara que utilizaremos para todo el correo que emitamos para nuestro servidor, para esto editamos la siguiente línea:

MASQUERADE_AS(`cluster.com.ec')dnl

Luego de realizar todas las modificaciones anteriores, procesamos el siguiente comando para generar el archivo */etc/sendmail.cf.*

[root@clusternodo1 root]#m4 /etc/mail/sendmail.mc > /etc/mail/sendmail.cf

/etc/mail/access

Abrimos ahora el archivo */etc/mail/access* y agregamos algunas líneas para definir quienes podrán hacer uso de nuestro servidor de correo para poder enviar mensajes.

[root@clusternodo1 mail]#vi access

```
# Check the /usr/share/doc/sendmail/README.cf file for a description
# of the format of this file. (search for access_db in that file)
# The /usr/share/doc/sendmail/README.cf is part of the sendmail-doc
# package.
#
# by default we allow relaying from localhost...
localhost.localdomain          RELAY
localhost                      RELAY
127.0.0.1                      RELAY
192.168.1.2                    RELAY
10.0.0.2                       RELAY
10.0.0.10                      RELAY
```

Al concluir, debemos también compilar este archivo, con la siguiente instrucción, para generar otro en forma de base de datos, el cual será utilizado por Sendmail.

[root@clusternodo1 mail]#makemap hash /etc/mail/access.db < /etc/mail/access
Culminada, la configuración, se reinicia el servicio sendmail del siguiente modo:

[root@clusternodo1 mail]#service sendmail restart

Habilitando el servicio IMAP

Habilitamos los servicios de manera automática e inmediata, ejecutando el siguiente comando:

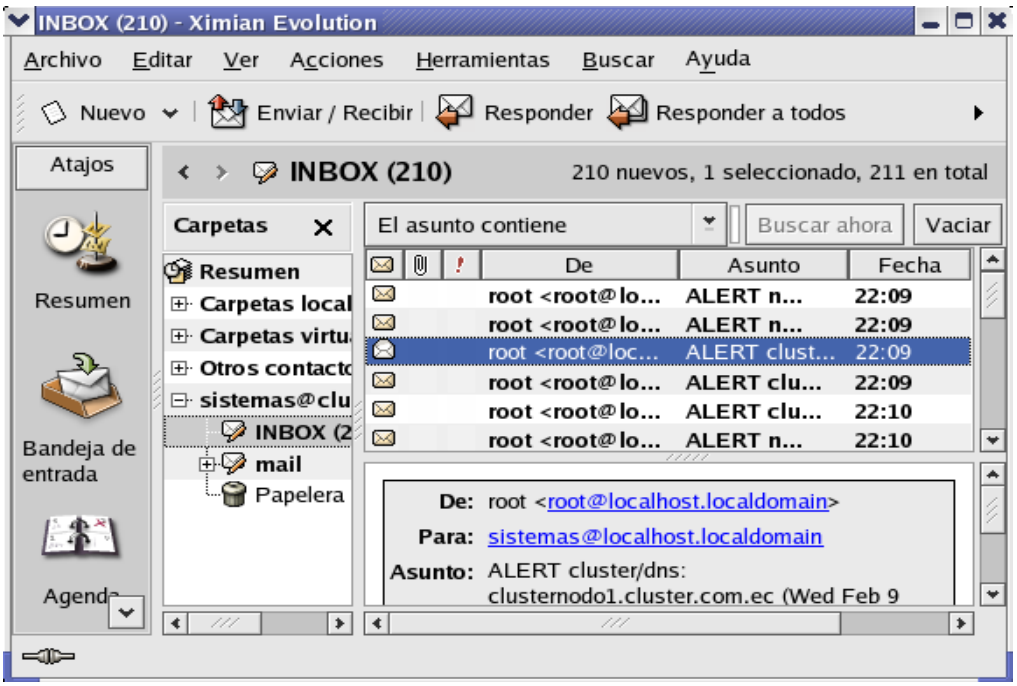
[root@clusternodo1 mail]#/sbin/chkconfig imap on
[root@clusternodo1 mail]#/sbin/chkconfig imaps on

Creación de cuentas de correo

Creamos las cuentas necesarias en el servidor de mail con el siguiente comando:

[root@clusternodo1 mail]#useradd sistemas -c " Departamento de Sistemas"
[root@clusternodo1 mail]#passwd sistemas
New Password: xxxxx
Retype New Password: xxxxx

Para probar que el servidor de mail está funcionando correctamente, configuramos el programa de correo Ximian Evolution1.4.5 para Linux Fedora Core I u Outlook Express para Windows, en las máquinas de los clientes con sus respectivas cuentas, y procedemos a enviar mails entre los usuarios.



4.2.2.4 Web Mail

Configuración

Ejecutamos el programa de configuración de la siguiente manera:

[root@clusternodo1 root]# /usr/share/squirrelmail/config/conf.pl

Ingresamos en el menú Organitation Preferences, para modificar los diferentes aspectos de la página principal de Webmail, tales como:

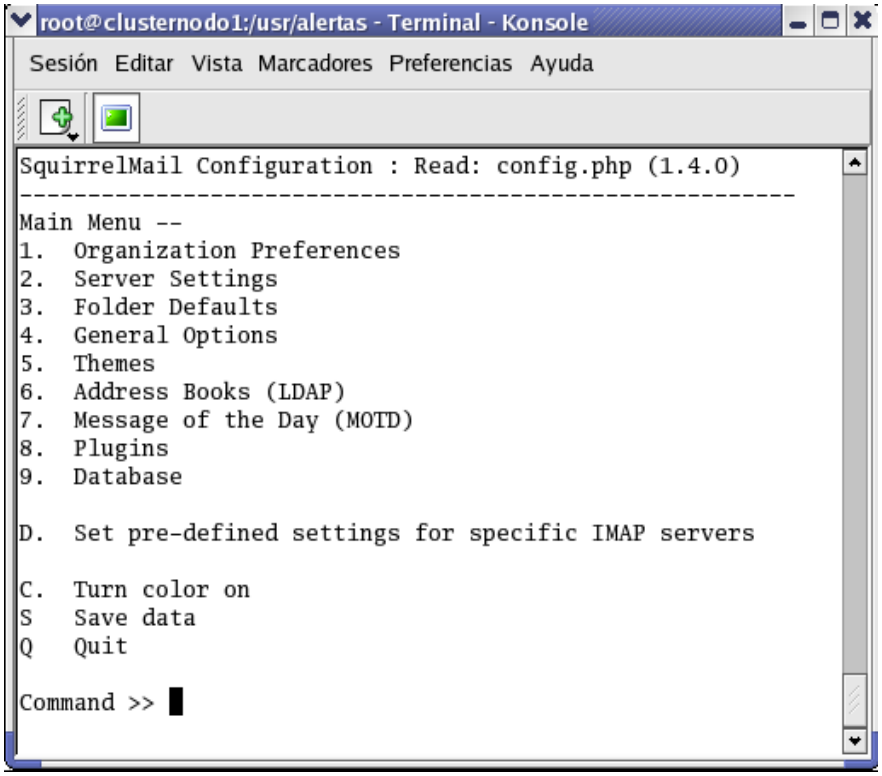
Organization Name:

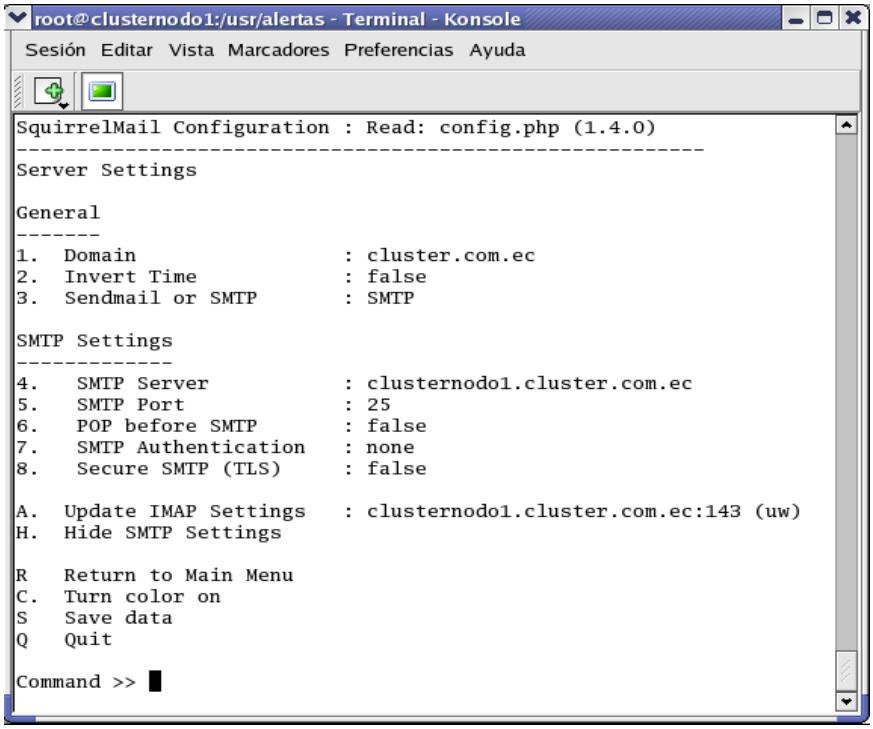
Organization Logo:

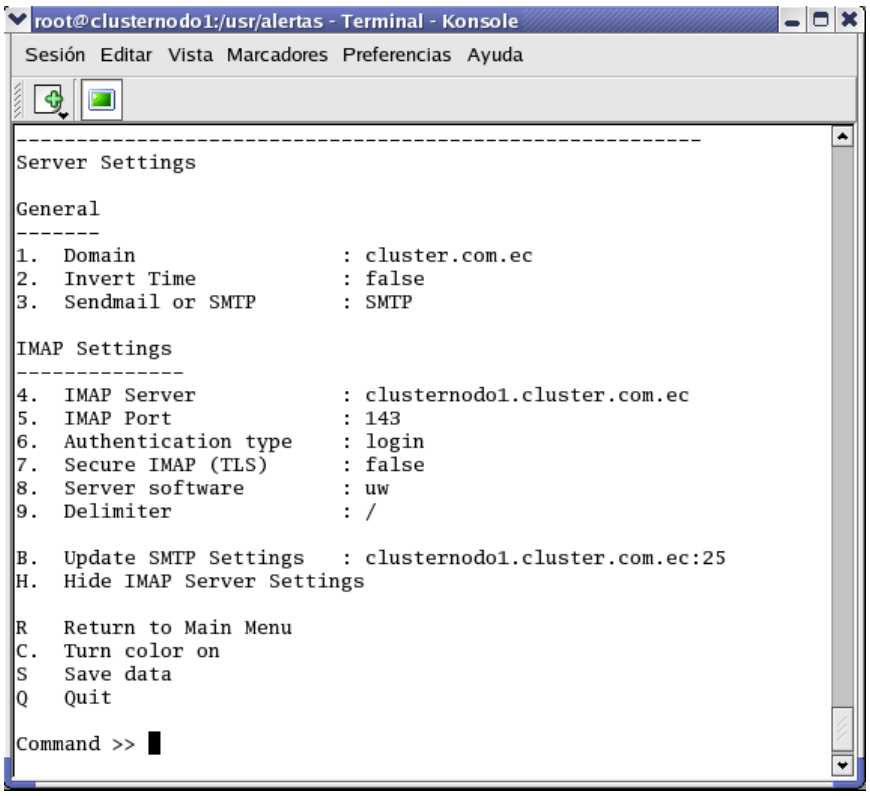
Organitation tittle:

Default Language:

Luego accedemos al menú ServerSettings, para configurar, el dominio, el tipo de servicio, e indicar que host es el servidor IMAP y cual el servidor SMTP, junto con el puerto por el cual escuchan, como se muestra en las figuras a continuación:




root@clusternodo1:/usr/alertas - Terminal - Konsole
Sesión Editar Vista Marcadores Preferencias Ayuda
SquirrelMail Configuration : Read: config.php (1.4.0)
Server Settings
General
1. Domain : cluster.com.ec
2. Invert Time : false
3. Sendmail or SMTP : SMTP
SMTP Settings
4. SMTP Server : clusternodo1.cluster.com.ec
5. SMTP Port : 25
6. POP before SMTP : false
7. SMTP Authentication : none
8. Secure SMTP (TLS) : false
A. Update IMAP Settings : clusternodo1.cluster.com.ec:143 (uw)
H. Hide SMTP Settings
R Return to Main Menu
C. Turn color on
S Save data
Q Quit
Command >>

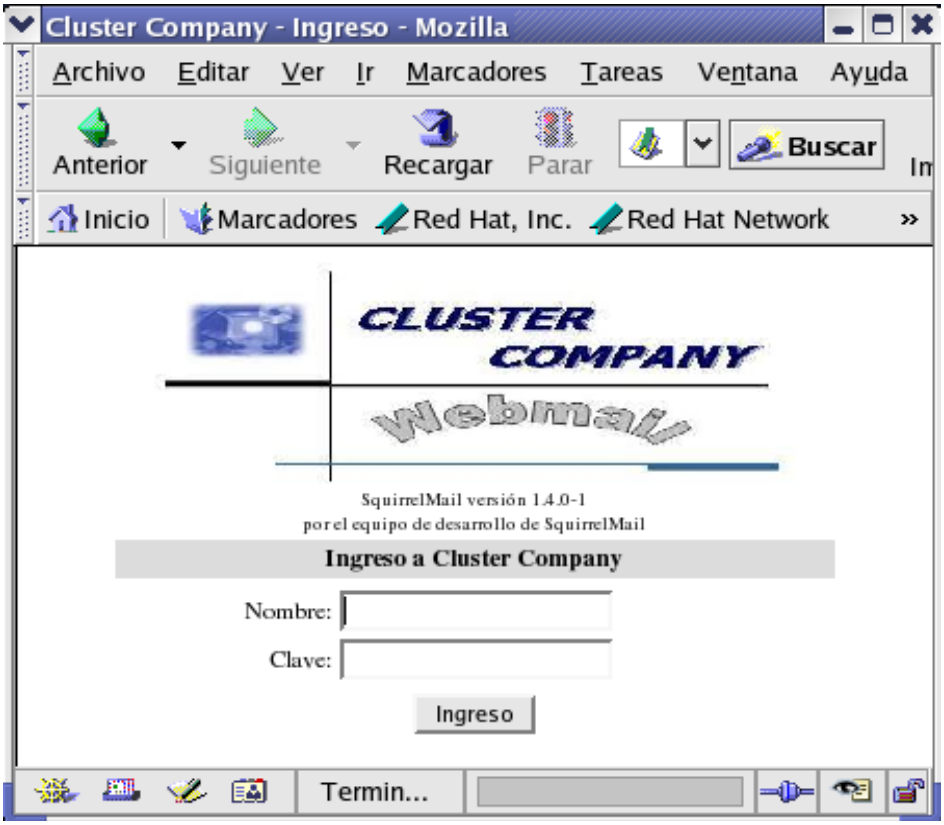

root@clusternodo1:/usr/alertas - Terminal - Konsole
Sesión Editar Vista Marcadores Preferencias Ayuda
Server Settings
General
1. Domain : cluster.com.ec
2. Invert Time : false
3. Sendmail or SMTP : SMTP
IMAP Settings
4. IMAP Server : clusternodo1.cluster.com.ec
5. IMAP Port : 143
6. Authentication type : login
7. Secure IMAP (TLS) : false
8. Server software : uw
9. Delimiter : /
B. Update SMTP Settings : clusternodo1.cluster.com.ec:25
H. Hide IMAP Server Settings
R Return to Main Menu
C. Turn color on
S Save data
Q Quit
Command >>

Ahora reiniciamos el servicio httpd.

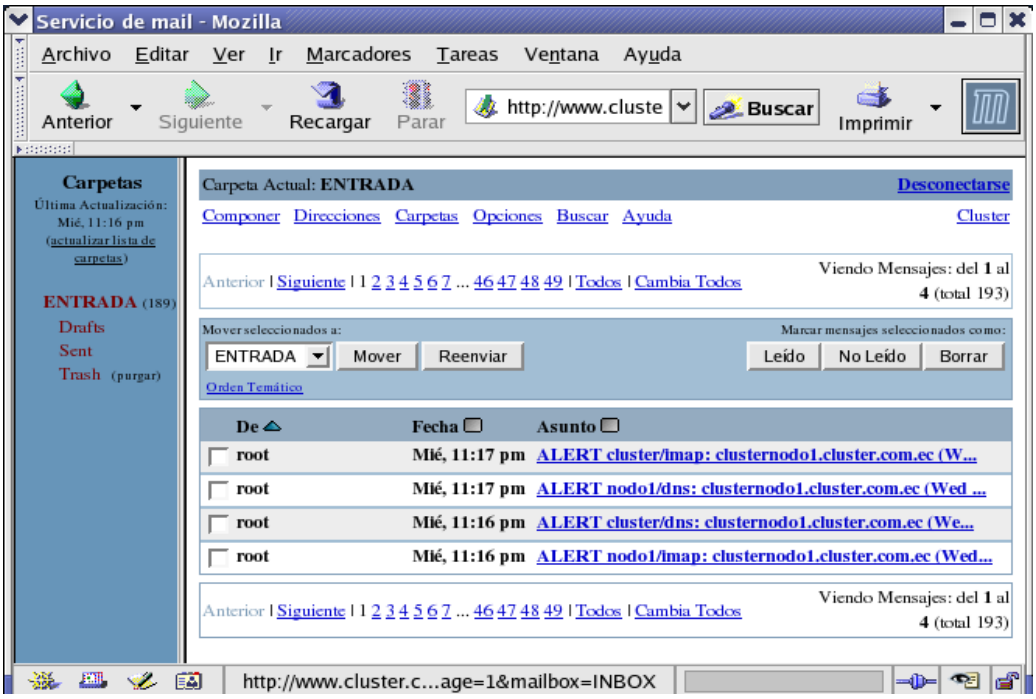
Para verificar que el Webmail está funcionando correctamente, escribimos en el browser:

http://www.cluster.com.ec/webmail

Y si todo está bien, se observará la página de presentación de Webmail que se modificó anteriormente, y ahora podremos logearnos con algún usuario, para enviar y recibir mails.



Cortéz Guerrero, Stanley
Galarza Rodas, Mónica



4.2.3 Herramientas de Alta Disponibilidad

4.2.3.1 Heartbeat

Como se habló en el capítulo II, Heartbeat es el programa que se encarga de monitorear si un equipo perteneciente al cluster está en funcionamiento o caído, a través de varios accesos.

Instalación

Heartbeat está disponible como paquetes RPM y en formato tar.gz. En este cluster se han instalado tanto el clusternodo1 como en el clusternodo2 los siguientes paquetes RPM, utilizando las siguientes instrucciones:

[root@clusternodo1 root]# rpm -i heartbeat-stonith-1.2.3-2.fr.c.1.i386.rpm
[root@clusternodo1 root]# rpm -i heartbeat-pils-1.2.3-2.fr.c.1.i386.rpm
[root@clusternodo1 root]# rpm -i heartbeat-1.2.3-2.fr.c.1.i386.rpm

Configuración

Los ficheros de configuración, se encuentran en el directorio */etc/ha.d*, que son creados luego de la instalación.
Existen 3 ficheros que se necesitan configurar en ambos nodos, antes de levantar heartbeat:

/etc/ha.d/authkeys

Editamos este fichero para dejar el mecanismo de autentificación en 1 de la siguiente manera:

[root@clusternodo1 root]# vi /etc/ha.d/authkeys

```
#/etc/ha.d/authkeys
auth 1
1 crc
#end /etc/ha.d/authkeys
```

/etc/ha.d/ha.cf

Este fichero es el archivo principal de configuración, el cual contiene varias opciones de rutas de accesos que se pueden utilizar. Las opciones que se han utilizado para esta práctica son:

deadtime 30	El nodo es pronunciado muerto después de 30 s.
serial /dev/ttyS0	Para realizar la monitorización a través del puerto serial(ttyS0)
bcast eth1	Realiza la monitorización por la interfaz eth1.
node clusternodo1	Nombres de los nodos que componen el cluster.
node clusternodo2	

[root@clusternodo1 root]# vi /etc/ha.d/ha.cf

```
debugfile       /var/log/ha-debug
logfile         /var/log/ha-log
logfacility     localo
deadtime        10
```

```
serial      /dev/ttyS0
bcast       eth1
node        clusternodo1
node        clusternodo2
```

/etc/ha.d/haresources

Editamos el fichero de la siguiente manera:

*[*root@clusternodo1 *root]# vi /etc/ha.d/haresources*

```
#masternode ip-address service-name
clusternodo1 10.0.0.100 named httpd sendmail
```

Esto define a clusternodo1 como nodo maestro, de forma que si clusternodo1 se va abajo, clusternodo2 tomará los servicios de named, httpd y sendmail a su cargo, pero cuando vuelva a estar disponible asumirá de nuevo los servicios.
El segundo elemento define la dirección IP que se debe asumir para los servicios, y el tercero el nombre de los servicios. Cuando una máquina asume el servicio ejecutará los mismos, mientras que la otra los dará de baja.

En el directorio /etc/ha.d/rc.d se encuentra un script llamado ip-request, que asignará las direcciones IP (fake) y otras tareas.

Una vez al tener configurado Heartbeat, procedemos a arrancar el mismo en ambas máquinas.

*[*root@clusternodo1 *root]#service heartbeat start*

Para verificar el correcto funcionamiento de heartbeat, instalaremos una página de inicio diferente en cada una de las máquinas que harán de servidor httpd.
Si todo se configuró correctamente clusternodo1 tendrá la IP 10.0.0.100 y será quién responda las peticiones named, httpd y sendmail. Si todo parece correcto le damos de baja a clusternodo1, en 10 segundos, clusternodo2 asumirá la dirección IP 10.0.0.100 y los servicios. Es decir el tiempo máximo de fuera de servicio será de 10 segundos.

4.2.3.2 Rsync

Para mantener la integridad de los datos se ha utilizado la herramienta Rsync, de esta manera tendremos una sincronización remota de los archivos que necesita el nodo maestro para brindar sus servicios, en el nodo backup. La ventaja de esta herramienta es que utiliza un mecanismo que permite copiar únicamente las modificaciones, y además las puede comprimir.

Configuración

En clusternodo2, se ha creado un script dentro de */sbin*, llamado *backupcliente.sh*, en el cual se ha definido, utilizando variables la dirección del nodo maestro, las carpetas y archivos remotos que necesitamos respaldar y los lugares donde se deben colocar estos respaldos .
Los parámetros del comando Rsync que se utilizaron para la configuración son los siguientes:

-e Esta opción permite escoger un programa de shell remoto para la comunicación entre la máquina remota y la máquina local.

-a Modo archivo (= -rlptDg)
- -r Recursivo
- -p Preservar Permisos
- -t Preservar Fecha

-u Solo actualiza, no sobrescribe ningún archivo

-v Modo verboso (-vv mas verboso). Esta opción incrementa la información sobre los archivos que serán transferidos. Por defecto, rsync trabaja silenciosamente

-z Comprimir (si lo admite el servidor).

Este script es el siguiente.

*[*root@clusternodo2 *root]# vi /sbin/backupcliente.sh*

```
#!/bin/bash
#script de sincronización del cliente
SERVIDOR=192.168.1.1 # Dirección IP del Servidor o Hostname
C_PROYECT_LOCAL=/var
#carpeta donde están los documentos www alojado localmente
C_PROYECT_LOCAL_USUARIOS=/etc
# carpeta de las cuentas de usuario
C_PROYECT_LOCAL_SHADOW=/etc
# carpeta de las claves de usuario
C_PROYECT_LOCAL_CORREO=/var/spool
# carpeta de  e-mail de usuario
C_PROYECT_LOCAL_HOME=/
# carpeta de usuarios del sistema
C_PROYECT_REMOTA=/var/www
# carpeta remota donde se aloja los documentos www
C_PROYECT_REMOTA_USUARIOS=/etc/passwd
# carpeta remota donde se están las cuentas de usuarios
C_PROYECT_REMOTA_SHADOW=/etc/shadow
# carpeta remota donde se están las claves de usuarios
C_PROYECT_REMOTA_CORREO=/var/spool/mail
# carpeta remota donde se están las e-mails de usuarios
C_PROYECT_REMOTA_HOME=/home
# carpeta remota donde se están los usuarios del sistema
# SITIO WEB
rsync -e ssh -auvzr --delete $SERVIDOR:$C_PROYECT_REMOTA
$C_PROYECT_LOCAL
# USUARIOS Y PASSWORD
rsync -e ssh -auvzr --delete $SERVIDOR:$C_PROYECT_REMOTA_USUARIOS
        $C_PROYECT_LOCAL_USUARIOS
rsync -e ssh -auvzr --delete $SERVIDOR:$C_PROYECT_REMOTA_SHADOW
                $C_PROYECT_LOCAL_SHADOW
# CORREOS DE USUARIOS
rsync -e ssh -auvzr --delete $SERVIDOR:$C_PROYECT_REMOTA_CORREO
                $C_PROYECT_LOCAL_CORREO
# HOME
rsync -e ssh -auvzr --delete $SERVIDOR:$C_PROYECT_REMOTA_HOME
                $C_PROYECT_LOCAL_HOME
```

Para automatizar la sincronización de los datos, añadimos este script en el demonio cron para que se ejectute cada minuto, y de esta manera tener el nodo backup lo más actualizado posible. Esto lo podemos hacer de la siguiente manera:

[root@clusternodo2 root]# crontab -e
** * * * * /sbin/backupcliente.sh*

Típicamente Rsync utiliza rsh por defecto, pero se ha configurado con el parámetro -e ssh, ya que éste es un shell que brinda alta seguridad.
Cuando el nodo local (clusternodo2) va a acceder mediante ssh al nodo remoto (clusternodo1) para realizar la sincronización de archivos, éste pide la autentificación del nodo local, por lo que se debe ingresar el password del nodo remoto. Esta operación se la debe automatizar para que se la realice de un forma transparente.

Esto se realiza de la siguiente manera:

Primero en el nodo local se crea la clave pública y privada con:

*[*root@clusternodo2 *root]#ssh-keygen -t rsa*

Ingresamos la ruta donde se va a agregar estas claves

/root/.ssh/id_rsa

Dejamos la passphrase en blanco.

Una vez generada, dentro del directorio /root/.ssh/, estarán los siguientes ficheros:

id_rsa
id_rsa.pub
known_hostsa

Aquí nos interesa el fichero *id_rsa.pub*, que es la clave pública. Se agregas el contenido de este fichero, en el archivo *authorized-keys* del nodo remoto de la siguiente manera:

*[*root@clusternodo1 *root]cat id_rsa.pub >> /root/.ssh/authorized_keys*

De esta manera Rsync podrá conectarse vía ssh sin necesidad de ingresar el password y realizándose así de forma transparente.

4.2.3.3 Mon

Mon nos ayuda en la monitorización de los servicios disponibles para enviar alertas, si ha ocurrido algún fallo en los éstos. Para este cluster hemos utilizado la versión mon-0-99-2.6. La siguiente configuración se ha realizado tanto en clusternodo1 como en clusternodo2.

Requerimientos

El demonio de mon utiliza Perl 5.n, donde n > 005_1. Mon también requiere que *.ph sean creados desde las cabeceras de archivos del sistema. Esto se realiza de la siguiente forma:

*[*root@clusternodo1 *root]*#cd /usr/include
*[*root@clusternodo1 *root]*#h2ph ***.h sys /**.h asm/*.h

Además se necesita los siguientes módulos para que el servidor funcione:

Time::Period
Time::HiRes
Convert-BER-1.3101.tar.gz
Mon::*
Digest::HMAC
Net::DNS

Estos módulos se instalan, ubicándose en el directorio en donde se encuentra cada módulo y ejecutando las siguientes instrucciones:

*[*root@clusternodo1 Period-1.20*]#perl Makefile.PL*
*[*root@clusternodo1 Period-1.20*]#make*
*[*root@clusternodo1 Period-1.20*]#make install*

También se requiere de la librería *Translation.pm,* la cual copiamos en el directorio*:*

/usr/lib/perl5/5.8.1

Instalación

Descomprimimos el paquete *mon-0.99.2.tar.gz* en el directorio */usr/lib/mon* .

/usr/lib/mon/auth.cf

Se edita el archivo */usr/lib/mon/auth.cf.* Este archivo controla, que usuario puede ejecutar que comando. Este archivo está formado por: command section y trap section.

*[*root@clusternodo1 mon*]#vi auth.cf*

```
# authentication file
# entries look like this:
# command: {user|all}[,user...]
# THE DEFAULT IT TO DENY ACCESS TO ALL IF THIS FILE
# DOES NOT EXIST, OR IF A COMMAND IS NOT DEFINED HERE
# command section
        command section
        ack:            all
        checkauth:      all
        clear:          all
        disable:        all
        dump:           all
        enable:         all
        get:            all
        list:           all
        loadstate:      all
        protid:         all
        quit:           all
        reload:         all
```

```
        reset:          all
        savestate:      all
        servertime:     all
        set:            all
        start:          all
        stop:           all
        term:           all
        test:           all
        version:        all
# trap section
# if no source hosts or users are defined, then do not
# accept traps
        trap section
#source_host  user    password
# allow from user "mon" from any host
# * mon monpassword
# allow from host 127.0.0.1 without requiring
# a valid username and password
        127.0.0.1 * *
        10.0.0.1 * *
        10.0.0.2 * *
        10.0.0.100 * *
        10.0.0.10 * *
        192.168.1.1 * *
        192.168.1.2 * *
```

La definición de comandos consiste de un comando, seguido por una coma, y seguido por una de usuarios que pueden ejecutar el comando separados por comas. Al especificar “all”, indicamos que todos los usuarios pueden ejecutar ese comando. Para denegar a un usuario, se especifica el nombre de usuario con “!”. Si la palabra "AUTH_ANY" es utilizada como nombre de usuario, ningún usuario autentificado estará permitido a ejecutar el comando.

En la sección trap se configura, que usuarios pueden enviar traes desde que hosts. Su sintaxis es: host de origen, espacio en blanco, el nombre de usuario, espacio en blanco y el password del usuario.

En nuestro caso hemos colocado todos los hosts de la red local y privada para que acepten traes de todos los usuarios, colocando "*" en el lugar de username y "*" en el lugar de password.

/etc/services

Agregamos las siguientes líneas en el archivo */etc/services*

mon 2583/tcp # MON
mon 2583/udp # MON traps

Esta configuración nos indica que se le asigna el puerto 2583 tanto TCP como UDP para el servicio mon.

/usr/lib/mon/mon.d

Instalamos los monitores que se encuentran en el directorio */usr/lib/mon/mon.d* con la siguiente instrucción:

*[*root@clusternodo1 mon.d*]#install -d /usr/lib/mon/mon.d*

/usr/lib/mon/mon.d/dns.monitor

Editamos el monitor */usr/lib/mon/mon.d/dns.monitor* de la siguiente manera, para que adopte la zona que cubre nuestro servidor dns:

my(@Zones) = ("cluster.com.ec");
my($Master) = "cluster.com.ec";

Configuración de alertas para reiniciar servicios

Ya que mon se ejecuta bajo el usuario daemon, que no tiene permisos para reiniciar servicios, utilizamos *sudo* con el siguiente fichero de configuración *sudoers,* en el cual añadimos las siguientes líneas en la sección de privilegios de usuarios, utilizando el editor visudo (editor de archivos sudoers):

*[*root@clusternodo1 mon.d*]#visudo*

```
# User privilege specification
root        ALL = (ALL) ALL
daemon      ALL = NOPASSWD:/etc/init.d/httpd restart
daemon      ALL = NOPASSWD:/etc/init.d/named restart
daemon      ALL = NOPASSWD:/etc/init.d/sendmail restart
```

Con estas líneas le damos permisos al usuario daemon de ejecutar con permisos de root, mediante sudo y sin necesidad de introducir ninguna contraseña, la orden de reiniciar los servicios de httpd, named y sendmail.

Los ficheros de las alertas quedarían así:

*[*root@clusternodo1 alert.d*]#vi apache.alert*

```
#!/bin/bash
#!/usr/bin/perl
/usr/bin/sudo -u root /etc/init.d/httpd restart
```

*[*root@clusternodo1 alert.d*]#vi dns.alert*

```
#!/bin/bash
#!/usr/bin/perl
/usr/bin/sudo -u root /etc/init.d/named restart
```

*[*root@clusternodo1 alert.d*]#vi sendmail.alert*

```
#!/bin/bash
#!/usr/bin/perl
/usr/bin/sudo -u root /etc/init.d/sendmail restart
/sbin/chkconfig imap on
/sbin/chkconfig imaps on
```

Cambiamos los permisos de los tres archivos anteriores de la siguiente manera:

*[*root@clusternodo1 alert.d*]#chmod 755 apache.alert*

*[*root@clusternodo1 alert.d*]#chmod 755 dns.alert*

*[*root@clusternodo1 alert.d*]#chmod 755 sendmail.alert*

/usr/lib/mon/mon.cf

Teniendo ya listo todos los monitores y alertas que serán utilizadas, se configura de la siguiente manera el archivo base del mon */usr/lib/mon/mon.cf*.

```
# "mon.cf" configuration for "mon".
# $Id: example.cf 1.1 Sat, 26 Aug 2000 15:22:34 -0400 trockij $
# This works with 0.38pre8
# global options
cfbasedir   = /usr/lib/mon
alertdir    = /usr/lib/mon/alert.d
mondir      = /usr/lib/mon/mon.d
maxprocs    = 20
histlength = 100
randstart = 60s
# authentication types:
#   getpwnam      standard Unix passwd, NOT for shadow passwords
#   shadow        Unix shadow passwords (not implemented)
#   userfile      "mon" user file
#authtype = getpwnam
# NB:  hostgroup and watch entries are terminated with a blank line (or
# end of file).  Don't forget the blank lines between them or you lose.
# group definitions (hostnames or IP addresses)
hostgroup                cluster                clusternodo1.cluster.com.ec
clusternodo2.cluster.com.ec
hostgroup nodo1 clusternodo1.cluster.com.ec
hostgroup nodo2 clusternodo2.cluster.com.ec
hostgroup workstations cliente.cluster.com.ec
# For the servers in building 1, monitor ping and telnet
# BOFH is on weekend call :)
watch cluster
    service ping
        description respuesta ping
        interval 1m
        monitor ping.monitor
        period wd {Mon-Sun}
                alert mail.alert sistemas
```

```
                alert file.alert /usr/alertas/alerta_ping
                alertevery 1m
    service dns
        description respuesta del servidor dns
        interval 1m
        monitor dns.monitor
        allow_empty_group
        period wd {Mon-Sun}
                alert mail.alert sistemas
                alert file.alert /usr/alertas/alerta_mail
                alert dns.alert
                alertevery 1m
    service http
        description respuesta del servidor http
        interval 1m
        monitor http.monitor
        period wd {Mon-Sun}
                alert mail.alert sistemas
                alert file.alert /usr/alertas/alerta_httpd
               alert apache.alert
               alertevery 1m
    service imap
        description respuesta del servidor imap
        interval 1m
            monitor imap.monitor
            period wd {Mon-Sun}
                    alert mail.alert sistemas
                    alert file.alert /usr/alertas/alerta_imap
                    alertevery 1m
watch nodo1
    service ping
            description respuesta ping
            interval 1m
            monitor ping.monitor
            period wd {Mon-Sun}
                    alert mail.alert sistemas
```

```
                alert file.alert /usr/alertas/alerta_ping
                alertevery 1m
    service dns
        description respuesta del servidor dns
        interval 1m
        monitor dns.monitor
        allow_empty_group
        period wd {Mon-Sun}
            alert mail.alert sistemas
            alert file.alert /usr/alertas/alerta_mail
            alert dns.alert
            alertevery 1m
    service http
        description respuesta del servidor http
        interval 1m
        monitor http.monitor
        period wd {Mon-Sun}
            alert mail.alert sistemas
            alert file.alert /usr/alertas/alerta_httpd
            alert apache.alert
            alertevery 1m
    service imap
        description respuesta del servidor imap
        interval 1m
        monitor imap.monitor
        period wd {Mon-Sun}
            alert mail.alert sistemas
            alert file.alert /usr/alertas/alerta_imap
            alertevery 1m
watch nodo2
    service ping
        description respuesta ping
        interval 1m
        monitor ping.monitor
        period wd {Mon-Sun}
            alert mail.alert sistemas
```

```
                alert file.alert /usr/alertas/alerta_ping
                alertevery 1m
    service dns
        description respuesta del servidor dns
        interval 1m
        monitor dns.monitor
        allow_empty_group
        period wd {Mon-Sun}
            alert mail.alert sistemas
            alert file.alert /usr/alertas/alerta_mail
            alert dns.alert
            alertevery 1m
    service http
        description respuesta del servidor http
        interval 1m
        monitor http.monitor
        period wd {Mon-Sun}
            alert mail.alert sistemas
            alert file.alert /usr/alertas/alerta_httpd
            alert apache.alert
            alertevery 1m
    service imap
        description respuesta del servidor imap
        interval 1m
        monitor imap.monitor
        period wd {Mon-Sun}
            alert mail.alert sistemas
            alert file.alert /usr/alertas/alerta_imap
            alertevery 1m
watch workstations
    service ping
        description respuesta Ping del cliente
        interval 1m
        monitor ping.monitor
        period wd {Sun-Sat}
            alert mail.alert sistemas
```

```
                alert file.alert /usr/alertas/alerta_cliente
                alertevery 1m
# This is the CrossView section
# The keyword are "alias", "service", "watch", "service", "items"
# You can see that you can make a crossview of everything
alias  pingCliente
    Este crossview es rojo si algún host en el grupo Cluster o
   Workstations falla la prueba de ping
     service Ping
                watch cluster service ping items
                watch workstations service ping items

alias Nodo1
    Este crossview es rojo si algún servicio del Nodo 1 tiene algún
    problema. Tu puedes ver cual es el problema
          service Ping
              watch cluster service ping items clusternodo1.cluster.com.ec
         service Dns
              watch cluster service dns items clusternodo1.cluster.com.ec
         service Http
             watch cluster  service http items clusternodo1.cluster.com.ec
        service Imap
            watch cluster  service imap items clusternodo1.cluster.com.ec

alias Nodo2
     Este crossview es rojo si algún servicio  del Nodo 2 tiene algún
     problema. Tu puedes ver cual  es el problema
        service Ping
            watch cluster service ping items clusternodo2.cluster.com.ec
       service Dns
            watch cluster service dns items clusternodo2.cluster.com.ec
      service Http
           watch cluster  service http items clusternodo2.cluster.com.ec
      service Imap
           watch cluster  service imap items clusternodo2.cluster.com.ec
```

Levantar el servicio de mon

Se da inicio al servicio de mon con la siguiente instrucción:

[root@clusternodo1 mon]#/usr/lib/mon/mon -f -c /usr/lib/mon/mon.cf -b /usr/lib/mon

Para verificar que todo está bien ejecutamos en la línea de comando lo siguiente:

*[*root@clusternodo1 mon*]#./clients/moncmd -s localhost list pids*

Y se obtendrá la respuesta de *list pids completed,* si todo está correcto.

Interfaz web de mon

Se ha utilizado la aplicación web Minotaur-0.03 para poder observar de manera gráfica la monitorización de servicios que se configuró en Mon. Para esto se realiza los siguientes pasos:

Copiar la página html de Minotaur en el DocumentRoot configurado en el Apache.

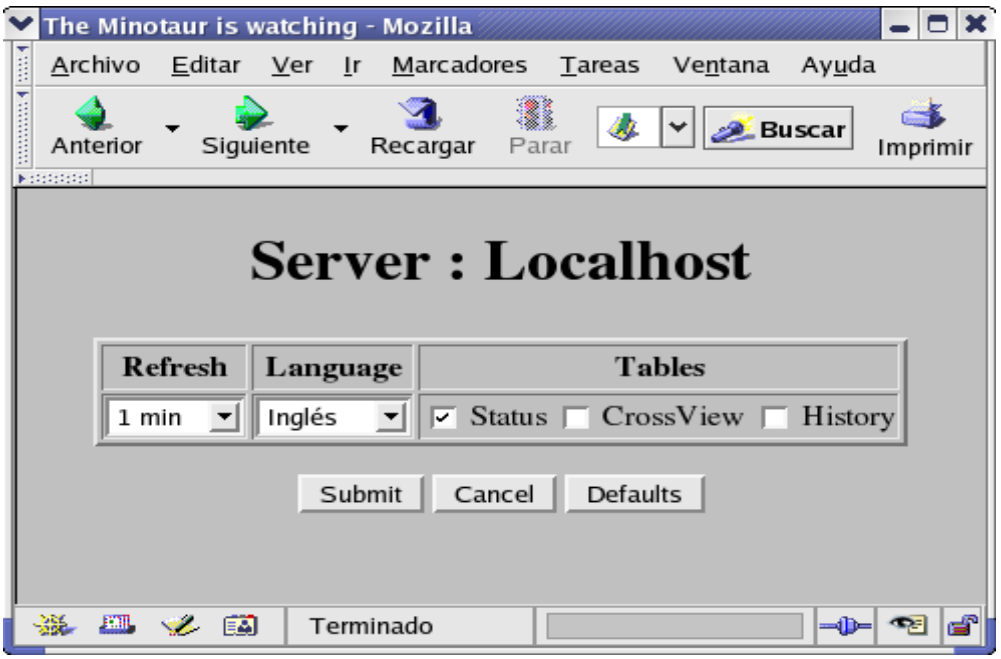
Copiar el script minotaur0.38.13.pl que viene en el paquete de Minotaur en la carpeta cgi-bin configurado en el Apache.

La presentación general consta de:

Refresh: en donde se debe seleccionar cada que tiempo se debe actualizar los resultados de la monitorización.

Language: Aquí podemos seleccionar el lenguaje de la interfaz a utilizar (English/Francais).

Tables: En estos checks boxes podemos seleccionar la información que deseamos observar, ya sea el estado de los servicios, cross view o un historial.



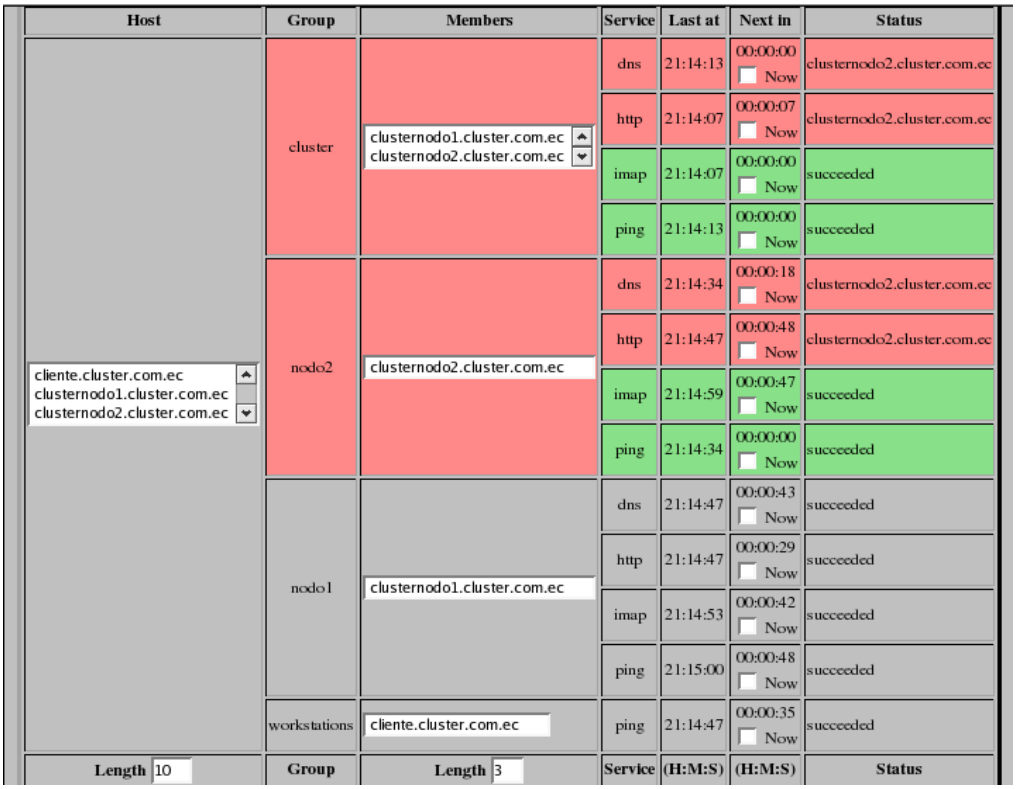
Status Table (Tabla de Estado)

Esta tabla está formada de 7 columnas que son:

Host: Esta columna contiene la lista de todos los hosts monitoreados.
Group: Contiene solo nombres de grupos de monitorización.
Members: Esta columna indica la lista de hosts del grupo.
Service: Indica los nombres de los servicios monitoreados,
Last at y Next in: La columna ¨Last at¨ nos muestra cuando fue por última vez testeado el servicio y la columna ¨Next in¨ indica cuanto falta para el próximo testeo.
Status: Esta columna muestra cual es el estado del servicio. Si el servicio está funcionando correctamente, esta celda contendrá ¨succeeded¨, caso contrario se observará ¨failed¨, seguido del host que falló.

Leyenda de Colores

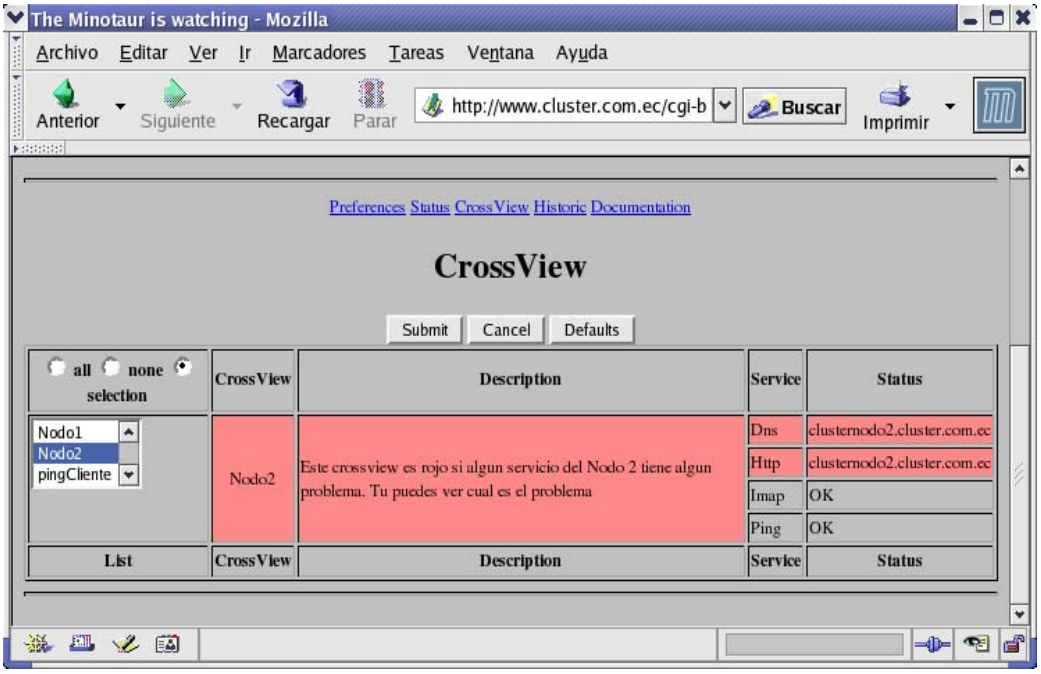
Cuando todos los servicios están funcionando correctamente, las celdas no tienen color.
Cuando un servicio es ¨down¨, éste y su grupo están en rojo, pero cuando un servicio del mismo grupo está levantado estará de color verde.
Cuando un servicio todavía no está testeado, se le mostrará de color amarillo.



Host	Group	Members	Service	Last at	Next in	Status
	cluster	clusternodo1.cluster.com.ec clusternodo2.cluster.com.ec	dns	21:14:13	00:00:00 Now	clusternodo2.cluster.com.ec
			http	21:14:07	00:00:07 Now	clusternodo2.cluster.com.ec
			imap	21:14:07	00:00:00 Now	succeeded
			ping	21:14:13	00:00:00 Now	succeeded
cliente.cluster.com.ec clusternodo1.cluster.com.ec clusternodo2.cluster.com.ec	nodo2	clusternodo2.cluster.com.ec	dns	21:14:34	00:00:18 Now	clusternodo2.cluster.com.ec
			http	21:14:47	00:00:48 Now	clusternodo2.cluster.com.ec
			imap	21:14:59	00:00:47 Now	succeeded
			ping	21:14:34	00:00:00 Now	succeeded
	nodo1	clusternodo1.cluster.com.ec	dns	21:14:47	00:00:43 Now	succeeded
			http	21:14:47	00:00:29 Now	succeeded
			imap	21:14:53	00:00:42 Now	succeeded
			ping	21:15:00	00:00:48 Now	succeeded
	workstations	cliente.cluster.com.ec	ping	21:14:47	00:00:35 Now	succeeded
Length 10	Group	Length 3	Service	(H:M:S)	(H:M:S)	Status

CrossView Table

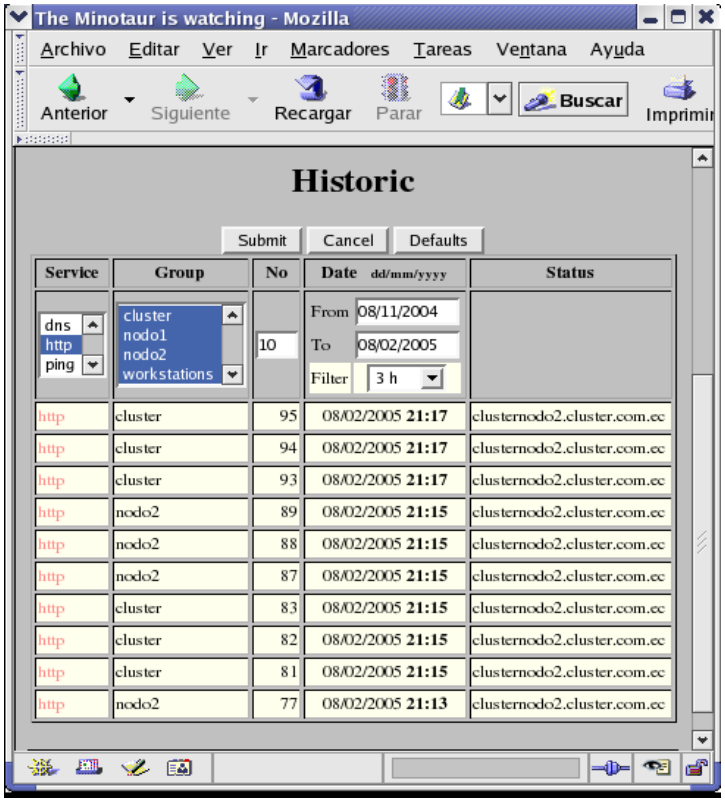
En esta tabla se puede monitorizar, los alias de los hosts que se han configurado.



Cortéz Guerrero, Stanley
Galarza Rodas, Mónica

Historic Table

Indica un historial de los servicios. Se puede escoger el número de líneas que se desea ver en el campo No, también se puede filtrar por fecha inicio, fecha de fin, horas, grupos y servicios.



4.3 Pruebas de la Implementación

Al terminar con la implementación y configuración de todo el hardware y software, hemos logrado construir un Cluster de Alta Disponibilidad Activo/Pasivo Linux que brinda los servicios de Web, Mail y DNS, los cuales fueron probados de la siguiente manera:

Desde nuestras estaciones cliente, por medio del browser ya sea Mozilla en Fedora Core I o Internet Explorer en Windows, hemos ingresado al sitio Web *www.cluster.com.ec* sin ningún inconveniente, dándose por entendido, que el servidor maestro Web y DNS (clusternodo1) funciona correctamente.

Luego se probó la disponibilidad de nuestro cluster, dando de baja (apagando) al servidor maestro. La prueba básica que se realizó, consistió en hacer ping a

www.cluster.com.ec desde las estaciones cliente y ver si la respuesta de ésta consulta al momento de darle de baja al servidor maestro, seguía siendo afirmativa, al realizar la prueba notamos que 5 segundo no hubo servicio ya el que nodo backup se encontraba levantando todos los servicios, luego la respuesta fue satisfactoria, dando por entendido que el servidor de respaldo (clusternodo2) se convirtió en el nuevo servidor maestro, brindado sin ningún inconveniente el servicio de DNS.

Después se realizó otra prueba muy similar, pero ahora conectándonos por medio del browser a www.cluster.com.ec, dimos de baja al nodo maestro (clusternodo1) y confirmamos que la conexión a www.cluster.com.ec siguió activa, pero ahora estábamos conectados al nodo de respaldo (clusternodo2), esto nos informaba que la sincronización de archivos entre los nodos del cluster estaba correcta y que los servicios Web y DNS iban a estar disponibles cuando el nodo maestro no este operando por cualquier motivo, además monitoreamos con la interfaz Web Minotaur, que todos los servicios eran brindados por clusternodo2 y no por clusternodo1.

Otra prueba que se realizó, fue comprobar que el servicio de Sendmail esté disponible cuando el nodo maestro no esté operando, pudiendo confirmar su correcto funcionamiento por medio del monitoreador que indicaba que clusternodo2 estaba brindando este servicio sin ningún problema, corroborando que el envío de mails entre usuarios estaba funcionando correctamente.

También se confirmó que el monitoreador de hardware (*heartbeat*) funciona correctamente, no solo apagando clusternodo1, sino también desconectando los cables de las interfaces de red y serial, simulando que han ocurrido problemas de hardware en éstos, con lo cual pudimos constatar que funciona muy bien el proceso de f*ake* que opera *heartbeat*.

Otro test que hemos realizado para probar la disponibilidad de nuestro cluster ha sido, detener manualmente los servicios HTTPD, NAMED y SENDMAIL, simulando que ha ocurrido un problema en éstos, ya sea por problemas de software, errores humanos, etc. Describimos a continuación como se realiza este test:

En clusternodo1 se detuvieron los servicios antes mencionados con la instrucción *service <servicio> stop*, pero a cabo de unos cortos segundos

podemos ver en la interfaz Web del monitoreador que los mismos servicios que les dimos de baja, volvieron a estar activos, confirmando que el monitoreador de servicios (*mon)* detectó el fallo y disparó las alertas configuradas tales como: los scripts para levantar los servicios, enviar mails a sistemas@cluster.com.ec informando el error de los servicios y crear un archivo de fallos. Esta prueba se realizó de la misma manera en clusternodo2 cuando estaba activo.

Y por último se realizó la siguiente prueba:

Cuando clusternodo1 para de funcionar por cualquier motivo, clusternodo2 se convierte en nodo maestro, pero cuando clusternodo1 vuelve a entrar en funcionamiento este se convierte, nuevamente en servidor maestro, brindando de esta manera los servicios de Web, mail y dns. Y clusternodo2 volvió a ser servidor backup. Esta operación se realizó satisfactoriamente gracias a la buena configuración y funcionamiento de los monitores de hardware (*heartbeat*) y de servicios (*mon*).

CONCLUSIONES

Al finalizar la investigación y la elaboración teórica y práctica de esta monografía, se ha llegado a la conclusión, que los Clusters de Alta Disponibilidad Linux son una buena opción para empresas que requieren de servidores robustos (Mainframes) que estén operativos ininterrumpidamente y no disponen de los recursos económicos para la implementación de los mismos, por esta razón los Clusters HA, se basan en la reutilización de PCs clónicas con software libre, quedando así de lado, el inconveniente de los recursos económicos de las empresa, para disponer de una arquitectura de servidores potente, escalable y robusta, teniendo varias opciones, desde las más sencillas que utilizan únicamente dos nodos, hasta las más complejas donde se podría disponer de infraestructuras de red redundantes (varios routers, varias salidas a Internet, varios cableados, varias tarjetas de red en cada equipo); dispositivos RAID por hardware; tecnología de discos compartidos GFS, etc.

Además existe gran variedad de software libre requerido para la implementación de un cluster, tales como Mon, Heartbeat, etc, que en conjunto puede llegar a ser tan potentes como otras aplicaciones propietarias, que existen en el mercado a un precio mucho mayor, proporcionando a las empresas un buen margen de beneficio.

Por esta razón, la inversión en clusters de alta disponibilidad es rentable, para servidores Web, de correo electrónico y cualquier aplicación que requiera un funcionamiento permanente, teniendo en cuenta, que una empresa que detiene su trabajo por fallo de su sistema, es una empresa que deja de percibir ingresos, es decir disminuyen sus utilidades, y por ende sus clientes.

El inconveniente que se ha encontrado, es el espacio físico que se requiere para el montaje de un cluster, ya que por lo menos para la instalación y montaje se necesitan una pantalla para cada uno de los nodos, pero una vez finalizada la instalación, se puede llevar a cabo el mantenimiento y administración desde una sola máquina mediante el uso de SSH.

Aunque existen ciertas áreas en las que el software libre todavía no está a la altura del software comercial, como por ejemplo en el campo de las Bases de Datos con soporte integrado para clustering, aún así, en estos casos el ahorro por utilizar software libre en el resto del cluster hace que Linux sea una buena alternativa al momento de planificar la infraestructura de servidores.

RECOMENDACIONES

Las siguientes recomendaciones servirán al momento de la implantación de un Cluster de Alta Disponibilidad.

Tener en cuenta a la hora de diseñar un cluster De Alta Disponibilidad, todos los componentes que éste requiere, al igual, cuáles son las herramientas más apropiadas, para el tipo de cluster a implementar, ya que estas deberán ser capaces de reaccionar a tiempo ante un fallo en cualquiera de los nodos.

Instalar las mismas versiones de software, en cada uno de los nodos que forman el cluster, para de esta manera evitar problemas de incompatibilidad.

Invertir parte del dinero ahorrado por no utilizar software propietario, en hardware de mejor capacidad o elementos externos que aseguren el buen funcionamiento del cluster.

Asegurar la perfecta sincronización entre el servidor primario y el de respaldo, para que las diferencias de contenidos entre ambos servidores en el momento del cambio de uno a otro sean mínimas.

Antes de instalar cualquier programa, se deben realizar pruebas para asegurar su buen funcionamiento, es recomendable, probar de forma aislada los programas y tecnologías a utilizar, para así conocer el rendimiento de cada una de las partes independientemente

BIBILIOGRAFIA

DOCUMENTACION PDF

AGUILAR, Vicente José	Clustering de Alta Disponibilidad bajo GNU/LINUX, Septiembre, 2001
CATALAN, Mikel	Nuevo Modelado de computación paralela con cluster Linux, Madrid, Septiembre, 2003.
ABIAN Y ACIBORG	Open Mosix Versión 1.0, Noviembre, 2002
PAREDES, Juan Pedro	Alta Disponibilidad para Linux
MARTINEZ, Marcos	Arquitecturas de Clustering de Disponibilidad y Escalabilidad (Linux Virtual Server), ACADE
PLAZA, Emilio José	Cluster Heterogéneo de Computadoras, Diciembre, 2003.
GARCIA, Rafael	Instalación y Configuración de un Cluster de Alta Disponibilidad bajo Linux, Abril, 2003.
MORALES, José María	Diseñando Sistemas de Alta Disponibilidad y Tolerantes a Fallos Versión 1.3, Madrid

DOCUMENTACION WEB

Archivos y Documentación del Kernel de Linux.
http://www.kernel.org

Información general de Linux
http://www.ilustrados.com

Información general de Linux
http://www.galeon.cpm/gnulinuxensupc/

Documentación de Cluster en Linux
http://cluster.fisica.uson.mx

Documentación diversa sobre clustering
http://www.hipacluster.org

Documentación de Disponibilidad en Linux
http://www.linux-ha.org/download/GettingStarted.html

Sitio Web del proyecto Linux-HA
http://linuxha.org

Documentación sobre la Alta Disponibilidad.
http://www.cs.mu.oz.au/~raj/tfcc/high-availability.html

Sistemas de Alta Disponibilidad Bajo Linux
http://www.linuxfocus.org

Sitio Web de Ultra Monkey
http://ultramonkey.sourceforge.net

Proyecto Linux Virtual Server
http://www.linuxvirtualserver.org

Download Módulos de Perl
http://search.cpan.org
http://www.perl.org

Documentación PDF sobre SSH
http://www.infocenter.gva.es

Información sobre RSYNC
http://www.unixmexico.org

Documentación sobre RSYNC
http://rsync.samba.org/

Documentación de como sincronizar con RSYNC
http://agila.org

Información de la herramienta Heartbeat
http://www.iso-net.com

Download de archivos y documentación de MON
ftp://ftp.kernel.org/pub/software/admin/mon

Documentación sobre Apache
http://www.24x7linux.com/documentacion/

Sitio oficial de Apache
http://www.apache.org

Información de Squirrelmail
http://bulma.net

Sitio oficial de Squirrelmail
http://www.squirrelmail.org

Documentación de Sendmail
http://www.linuxparatodos.net

Documentación de RAID en linux
http://www.gui.uva.es

REFERENCIAS

[1] Abian y Aciborg, Open Mosix Versión 1.0.
Pg. 1

[2] Lizárraga, Carlos y Méndez Antonio, Cluster de Linux, Universidad de Sonora

http://www.fisica.uson.mx/carlos/LinuxClusters/clusters_de_Linux.htm

[3] García, Rafael, Instalación y Configuración de un Cluster de Alta Disponibilidad bajo Linux.
Pg. 1

[4] Aguilar, Vicente José, Clustering de Alta Disponibilidad bajo GNU/LINUX
Pg. 66

[5] Aguilar, Vicente José, Clustering de Alta Disponibilidad bajo GNU/LINUX
Pg. 1.

[6] Catalan, Mikel, Clusters HA
http://www.hispafuentes.com/hf-doc/temas/herramien/clustering/node17_ct.html