ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE INGENIERÍA

CONFIGURACIÓN DE UN *CLUSTER* DE ALTA DISPONIBILIDAD Y BALANCEO DE CARGA EN LINUX PARA SATISFACER GRAN DEMANDA WEB Y SERVICIOS DE RESOLUCIÓN DE NOMBRES.

PROYECTO PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN ELECTRÒNICA Y REDES DE INFORMACIÓN

BUSTOS BURBANO ANDRÉS GUSTAVO

DIRECTOR: Ing. Xavier Armendáriz.

Quito, MARZO 2007

DECLARACIÓN

Yo Andrés Gustavo Bustos Burbano declaro que el trabajo aquí descrito es de mi autoría; que no ha sido previamente presentada para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

La Escuela Politécnica Nacional, puede hacer uso de los derechos correspondientes a este trabajo, según lo establecido por la Ley de Propiedad Intelectual, por su Reglamento y por la normatividad institucional vigente.

_____

ANDRÈS GUSTAVO BUSTOS BURBANO

CERTIFICACIÓN

Certifico que el presente trabajo fue desarrollado por Andrés Gustavo Bustos Burbano, bajo mi supervisión.

_____

Ing. Xavier Armendáriz

DIRECTOR DEL PROYECTO

AGRADECIMIENTO

A Dios, quien es el ser que me da fortaleza y sabiduría para culminar esta meta, a mis padres y hermanas, gracias por el apoyo incondicional y por haber creído en mí, a Gina Córdova, gracias por la paciencia y la comprensión.

A mis maestros universitarios, por su compromiso de formar profesionales de gran calidad, pero lo mas importante gracias por su amistad.

No podía olvidarme de mis valiosos amigos dentro y fuera de mi vida universitaria, pero de manera especial quiero dar las gracias a Diego Fernández, David Bravo, Fernando Moya, quienes estuvieron junto a mí cuando mas necesitaba.

A los amigos que están lejos, a quienes gratamente recuerdo.

Sinceramente,

_____

ANDRÉS GUSTAVO BUSTOS BURBANO

DEDICATORIA

A Britthany Anahí, mi hermosa hija, quien se ha convertido en mi motivo y razón de vida para luchar y superarme cada día, además a mi familia quien siempre confió en mí y me dio la mano cuando más lo necesitaba.

_____

ANDRÉS GUSTAVO BUSTOS BURBANO

CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

RESUMEN

El presente proyecto de titulación analiza una técnica para brindar alta disponibilidad por medio de la denominada redundancia mediante la utilización de varios servidores instalados en lugar de uno sólo, los cuales tengan la capacidad de trabajar en paralelo y de asumir las caídas de algunos de sus compañeros; por otra parte esta técnica permitirá la adición y eliminación de servidores al grupo según las necesidades. A esta técnica se la denomina *clusters* de servidores (granja de servidores).

El proyecto de titulación se encuentra destinado también al análisis del balanceo de carga, con lo cual surge el concepto de "*cluster* de servidores virtuales", el cual permite que un conjunto de servidores de red (*cluster*) compartan la carga de trabajo y el tráfico de diferentes clientes, haciendo que el servicio sea transparente para el cliente dando la ilusión de que es único servidor. Al balancear la carga de trabajo en un conjunto de servidores se mejora el tiempo de acceso y la confiabilidad. Además como es un conjunto de servidores el que atiende el trabajo la caída de uno de ellos no ocasiona una caída total del sistema. Este tipo de servicio es de gran valor para compañías que trabajan con grandes volúmenes de tráfico y trabajo en sus servidores: Web, correo electrónico y bases de datos.

Los objetivos del presente proyecto se desarrollan a lo largo de cuatro capítulos, con el siguiente contenido:

En el primer capítulo se realiza un breve estudio de las tecnologías paralelas, tecnología *cluster,* sus componentes de hardware y software, características, clasificación, y arquitectura de funcionamiento. Además un breve estudio relacionado a la administración y planificación de tareas.

En el segundo capítulo se realiza un breve estudio de los sistemas de archivos existentes, estudio de herramientas para implementar *cluster* de alta

disponibilidad y balanceo de carga, en especial HA OSCAR y LVS respectivamente, además se mencionan algunas herramientas de administración y planificación de tareas.

En el tercer capítulo se realiza un breve estudio de otras implementaciones realizadas, se especifican los requerimientos de hardware y software necesarios para la configuración del *cluster,* se configuró el *cluster* con las herramientas de código abierto como son HA OSCAR (*Open Source Cluster Aplication Resources High Avaibility*) y LVS (*Linux Virtual Server*) para alta disponibilidad y balanceo de carga respectivamente, se diseñó la aplicación Web e implementación de servicios (resolución de nombres y servicio de correo electrónico - *Postfix*). Una vez concluida la fase de instalación y configuración, se realizan las pruebas del caso utilizando el *benchmark ECperf*, el cual es el encargado de generar un alto tráfico http con la finalidad de verificar el funcionamiento real del *cluster* configurado, para finalizar el capítulo se compara los resultados obtenidos con otras tecnologías de servidores Web, correo electrónico, de resolución de nombres, tecnologías SAN (Storage Area Network) y NAS (Network Attached Storage).

Para finalizar el presente trabajo en el capítulo cuatro se presentan las conclusiones que arrojó el proceso de estudio, instalación y configuración del *cluster,* así como se ofrecen recomendaciones relacionadas al proyecto.

PRESENTACIÓN

Los altos costos que representa montar una infraestructura de alta disponibilidad y balanceo de carga ha sido un limitante para empresas que requieren de estas características para poder funcionar adecuadamente y evitar tener pérdidas considerables en su accionar cotidiano.

El presente proyecto de titulación busca solucionar el problema del inminente crecimiento del Internet el cual genera un aumento de tráfico en la red de forma radical y con él la carga de trabajo que ha de ser soportada por los servidores, especialmente por los servidores Web. Para solucionar este problema se han visto las siguientes alternativas: un servidor basado en una máquina de altas prestaciones, que a largo plazo probablemente quede obsoleta por el crecimiento de la carga, o bien se encamina la solución a la utilización de la tecnología de *clusters* para mantener un *cluster* de servidores ya sea de balanceo de carga y servicios de alta disponibilidad.

El servidor de balanceo de carga o director permitirá compartir la carga de trabajo y de tráfico de sus clientes. Al balancear la carga de tráfico en un arreglo de servidores, mejora el tiempo de acceso y confiabilidad. Además al ser un conjunto de servidores el que atiende el trabajo, la falla de uno de ellos no ocasiona una falla total y la alta disponibilidad va a contribuir en la solución de dicho problema.
Las configuraciones de alta disponibilidad son una parte importante encontrada en los *clusters* la cual trata del mantenimiento de servidores, para que exista un servicio de escucha y si se pierde la conexión o si ocurre un fallo en el mismo, exista un respaldo que inmediatamente pueda suplir las necesidades de balanceo de carga.

La utilización de *clusters* permite obtener una gran capacidad computacional tales como incremento de velocidad de procesamiento, incremento del número de transacciones o velocidad de respuesta e incremento de confiabilidad, y están

orientados a conjuntos o conglomerados de computadoras que trabajan de forma coordinada para dar la ilusión de un único sistema.

Dichos *clusters* están construidos utilizando componentes de hardware comunes, software libre e interconexión de redes de alta velocidad, en la actualidad son importantes en la solución de problemas en ciencias, ingeniería y aplicaciones comerciales.

Las configuraciones de alta disponibilidad son una parte importante encontrada en los *clusters* la cual trata del mantenimiento de servidores, para que exista un servicio de escucha y si se pierde la conexión o si ocurre un fallo en el mismo, exista un respaldo que inmediatamente pueda suplir las necesidades de balanceo de carga.

Otro punto importante es la redundancia, la cual permite que los servidores virtuales actúen entre ellos como respaldos de la información que éstos sirven. La flexibilidad y robustez que proporcionan este tipo de *clusters*, los hacen necesarios en ambientes de intercambio masivo de información, almacenamiento de datos sensibles y donde sea necesaria una disponibilidad continua del servicio ofrecido.

Con la creación de dichos *clusters* se va a obtener muchas ventajas no necesariamente técnicas sino también económicas mejorando la relación costo/beneficio de las inversiones en infraestructuras y acceder a campos que resultaban prohibitivos por los elevados montos de inversión involucrados anteriormente.

CAPÍTULO 1
INTRODUCCIÓN A *CLUSTERS*

1.1 REVISIÓN DE TECNOLOGÍAS PARALELAS.[1][2][3][4][5]

Durante varios años se han realizado diversos intentos para poder clasificar las arquitecturas computacionales. A pesar de que ninguna clasificación es perfecta, la más ampliamente utilizada es la taxonomía propuesta por Michael Flynn en el año de 1972. La taxonomía de Flynn considera dos factores: el número de instrucciones y el número de flujos de datos que maneja el procesador. Una máquina puede tener uno o múltiples flujos de datos y puede trabajar con estos datos en uno o varios procesadores, dando lugar a cuatro posibles combinaciones:

- SISD (*Single Instruction stream, Single Data stream*). Máquinas que poseen un único procesador.
- SIMD (*Single Instruction stream, Multiple Data streams*). Máquinas que poseen un punto de control único, ejecuta la misma instrucción de forma simultánea sobre múltiples valores de datos. Estas máquinas están clasificadas en arreglos de procesadores, procesadores vectoriales y arreglos sistólicos.
- MISD (*Multiple Instruction streams, Single Data stream*). Estas máquinas tienen múltiples flujos de instrucciones que operan sobre el mismo flujo de datos.
- MIMD (*Multiple Instruction streams, Multiple Data streams*). Estas máquinas emplean múltiples puntos de control los cuales poseen flujos de instrucciones y datos independientes. Se consideran MIMD a los sistemas multiprocesadores y paralelos.

Los computadores SIMD son simples de diseñar comparados con las máquinas MIMD pero se consideran menos flexibles, debido a que todos los

multiprocesadores SIMD deben ejecutar la misma instrucción de forma simultánea; por ejemplo la ejecución de un condicional podría ser costosa.

La taxonomía de Flynn no es apropiada en varias áreas: una de ellas es que existen muy pocas aplicaciones (si es que existen) para las máquinas MISD. Segundo Flynn asume que el paralelismo es homogéneo. Una colección de procesadores puede ser homogénea o heterogénea.

Otro problema en la taxonomía de Flynn es los sistemas MIMD. Una arquitectura multiprocesador se encuentra dentro de esta clasificación sin considerar como está conectado el procesador o como se encuentra distribuida su memoria. Existen varios intentos para refinar la clasificación MIMD.

Varios cambios sugieren subdividir la clasificación de procesadores de acuerdo a como están conectados: vía bus o vía conmutación. Los sistemas de memoria compartida son aquellos donde el procesador tiene acceso a la memoria global y se comunican a través de variables compartidas, de igual forma que los procesos en un sistema de procesador único. Como consecuencia todos los procesadores se comunican mediante el paso de mensajes lo cual puede ser costoso e ineficiente. El problema que varios autores tienen para realizar una clasificación del hardware se basa en que los términos de memoria compartida y paso de mensajes son modelos de programación y no modelos de hardware.

Los paradigmas más grandes de arquitecturas paralelas son SMP (*Symmetric MultiProcessors*) y MPP (*Massive Parallel Processors*) y se consideran arquitecturas MIMD pero difieren uno del otro por cómo utilizan la memoria. Las máquinas SMP comparten la memoria mientras las MPP no lo hacen. Las máquinas MPP típicamente albergan miles de CPUs en un gran contenedor conectados a cientos de GB de memoria. El término MPP describía el acoplamiento de multiprocesadores SIMD; actualmente este término es utilizado para arquitecturas paralelas que poseen nodos con memoria privada, cada uno de los cuales tiene la habilidad de comunicarse uno con otro a través de una red. La forma más sencilla de diferenciar un sistema MPP de un SMP es la siguiente:

- MPP = varios procesadores + memoria distribuida + comunicación vía red.
- SMP = pocos procesadores + memoria compartida + comunicación vía memoria.

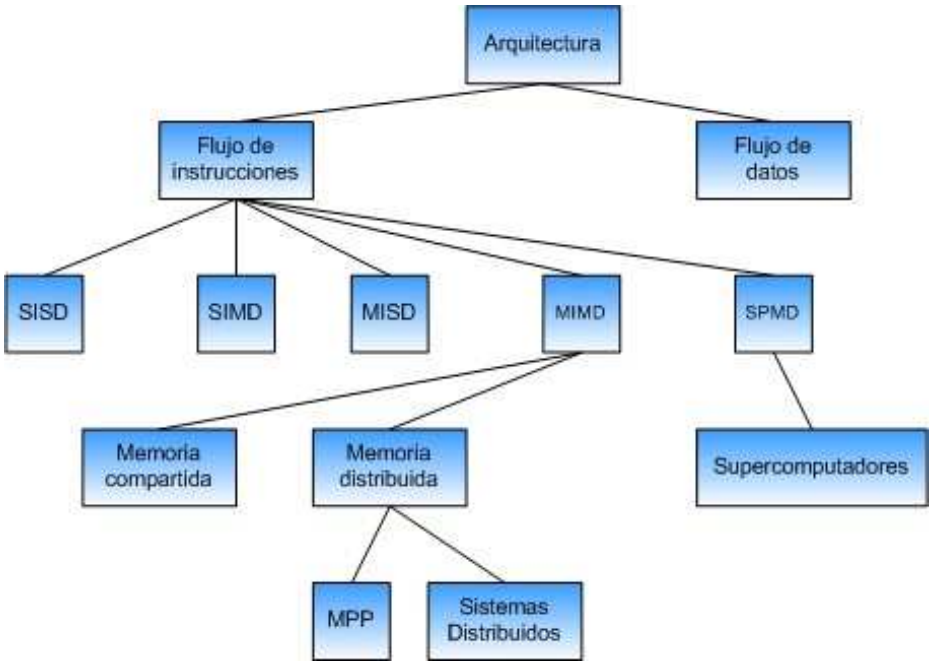


Figura 1-1. Taxonomia de arquitectura de computadores[1].

En los sistemas SMP el tiempo de acceso a memoria es uniforme por tal motivo se les conoce como arquitecturas UMA (*Uniform Memory Access*).

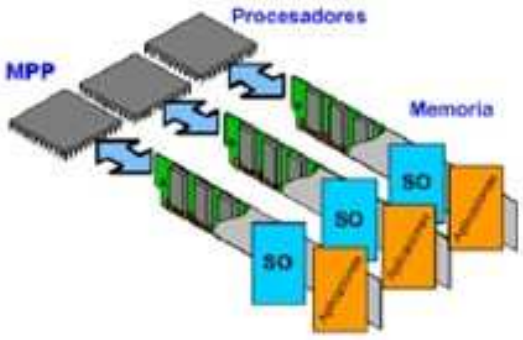


Figura 1-2. Arquitectura MPP [2]

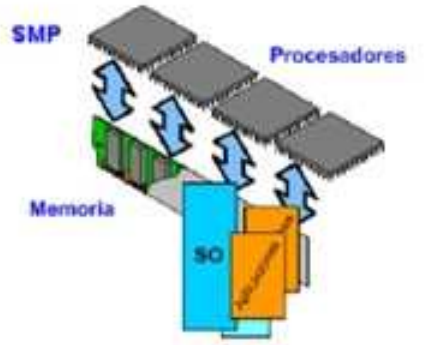


Figura 1-3. Arquitectura SMP[2]

La computación distribuida es otro ejemplo de arquitectura MIMD. La computación distribuida se define como un conjunto de computadores que trabajan de forma colectiva colaborando para resolver un problema. Esta colaboración puede darse de diversas maneras.

Una red de estaciones de trabajo[1] o NOW (*Network Of Workstations*) es una colección de estaciones de trabajo distribuidas que trabajan en paralelo solo si los nodos no se utilizan como estaciones de trabajo normales. Las NOWs son sistemas heterogéneos con diferentes procesadores y software (sistema operativo) que se comunican a través de Internet o una red WAN (*Wide Area Network*). Los usuarios deben establecer una conexión apropiada a la red antes de realizar un cómputo paralelo. A los NOWs también se les conoce como *grids*[2].

Un *cluster* de estaciones de trabajo o COW (*Cluster Of Workstations*) es una colección similar a la NOW pero requiere que una sola entidad o nodo que esté a

[1] Hoy en día ya no existe una diferencia entre un computador personal y una estación de trabajo o *workstation* debido a las mejoras que día a día se inovan en CPU, tiempo de acceso a memoria, incorporación de buses de alta velocidad para dispositivos de almacenamiento e interconexión.

[2] "La computación *grid* crea organizaciones virtuales que permiten compartir recursos distribuidos geográficamente, asumiendo la ausencia de una localidad central o un control central, cuyo objetivo es el permitir resolver problemas complejos.

cargo de las operaciones y el control. Los nodos de COW poseen software común y el usuario puede acceder a todos los nodos.

Un *cluster* dedicado como computador paralelo o DCPC (*Dedicated Cluster Parallel Computer*) es una colección de estaciones de trabajo específicamente agrupados para trabajar sobre un cómputo paralelo dado. Las estaciones de trabajo poseen software y sistemas de archivos en común, son administrados por una sola entidad o nodo y se comunican a través de Internet. Al igual que en NOW los nodos no se pueden utilizar como estaciones de trabajo normales.

Una pila de PCs o PoPC (*Pile of PCs*) es un *cluster* de hardware heterogéneo utilizado para construir un sistema paralelo. Los sistemas DCPC poseen pocos componentes los cuales son veloces pero costosos en precio. Los sistemas PoPC utilizan un gran número de nodos lentos pero menos costosos que los del sistema DCPC denominados *commodities*.

El proyecto *Beowulf* iniciado en el año de 1994 por Thomas Sterling y Donald Becker del *Goddard Space Flight Center* utiliza una arquitectura PoPC la cual acopla varias plataformas de hardware con un software especialmente diseñado para dar la ilusión de ser un solo computador, convirtiéndose en una máquina paralela. Los nodos *Beowulf* se interconectan a través de una red privada o LAN (*Local Area Network*).

La taxonomía de Flynn incluye recientemente a los sistemas SPMD (*Single program Multiple Data*). Un SPMD consiste de multiprocesadores cada uno con su propio conjunto de datos y memoria de programa. El mismo programa se ejecuta sobre cada procesador con la ayuda de sincronismo de varios puntos de control global. Cada procesador carga el mismo programa sin embargo cada procesador ejecuta diferentes instrucciones.

A su vez aparecieron los sistemas de Acceso No Uniforme a Memoria o NUMA (*Non-Uniform Memory Access*) incorporando mejoras dando origen a los sistemas CC-NUMA (*Cache Coherence Non-Uniform Memory Access*). Para reducir el

tiempo de acceso a memoria cada procesador NUMA mantiene memoria caché privada. Debido a que la memoria está distribuida físicamente, no se garantiza que las operaciones de acceso a los datos siempre se satisfagan al mismo tiempo. El término coherencia en caché se refiere a que cualquier variable que vaya a ser usada tenga un valor consistente en todos los procesadores.

1.2 FUNDAMENTOS GENERALES DE *CLUSTER*S [6]

El rápido crecimiento del Internet, el acelerado incremento del comercio electrónico, aplicaciones y la convergencia de servicios hacen indiscutible la importancia y la necesidad de que empresas (grandes o pequeñas) y sus departamentos de IT[3] piensen en garantizar el funcionamiento de su infraestructura: servicios, aplicaciones y comunicaciones al mayor grado de confiabilidad, integridad, disponibilidad y calidad de servicio. Las necesidades empresariales y de los departamentos de IT se traducen en servicios ininterrumpidos y sin errores 24-7[4] de *alta disponibilidad.*

Empresas como SUN e IBM fabrican sistemas informáticos redundantes, los cuales constan de grandes máquinas multiprocesador, arreglos RAID[5], fuentes de alimentación redundante, varios controladores de disco; entre otras. Estos sistemas de redundancia garantizan el funcionamiento del sistema hasta el momento en que su capacidad llega al límite; de esta manera sistemas de este tipo quedan obsoletos. La obsolescencia y el alto precio son una de las principales desventajas de estos sistemas informáticos.

Es por ello que la existencia de los *clusters* juegan papel muy importante en la solución de problemas de las ciencias, las ingenierías y en el desarrollo y ejecución de muchas aplicaciones comerciales, la utilización de componentes de hardware comunes, software libre e interconexión de redes de alta velocidad lo hace muy atractivo.

---

[3] IT: *Information Technologies.*

[4] 24-7 Referido a 24 horas al día, 7 días a la semana y los 365 días del año.

[5] RAID (*Redundant Array of Inexpensive Disks*) – Ver Capitulo 2, Sección 1.1.5.1.1.

La utilización de *clusters* permite obtener una gran capacidad computacional tal como incremento de velocidad de procesamiento, incremento del número de transacciones o velocidad de respuesta e incremento de confiabilidad, y están orientados a conjuntos o conglomerados de computadoras que trabajan de forma coordinada para dar la ilusión de un único sistema, donde la falla de una de estas evita tener una falla general en el sistema, ya que existen el resto de computadoras que suplen su ausencia.

1.2.1 CONCEPTO [7][8]

Los *clusters* de equipos vienen siendo utilizados por más de diez años. *G. Pfister* uno de los pioneros de la Tecnología de conglomerados mencionó que un *cluster* es: "…un sistema paralelo o distribuido que consta de una colección de equipos completos conectados entre sí que se utiliza como un recurso de equipos único y unificado…" [6].

Un *cluster* es un sistema que posee una arquitectura distribuida, formada por un grupo de equipos independientes, que ejecutan acciones de manera conjunta y que aparecen como un único recurso computacional ante aplicaciones y clientes.

Un *cluster* adquiere gran importancia para impulsar y garantizar escalabilidad, confiabilidad y disponibilidad ya que equipos de mediana potencia interconectados entre sí por redes de alta velocidad forman un arreglo de nodos redundantes. Si uno de los nodos falla los mecanismos existentes permitirán reemplazarlo y hacer que el sistema en general siga en funcionamiento.

En síntesis, un *cluster* es un sistema de varios computadores o nodos interconectados entre sí a través de dispositivos de alta velocidad dando la ilusión

---

[6] Para mayor información recurrir al texto "*In Search of Clusters*, Gregory F. Pfister, Prentice-Hall, 1998.". ó Visitar sitio Web: http://www.microsoft.com/latam/technet/articulos/windows2k/clustsrv/.

de ser un solo equipo, y cuyos componentes actúan en conjunto aprovechando su poder de cómputo para resolver ciertos problemas que se presentan.

Los supercomputadores tradicionales poseen costos excesivamente elevados cuya capacidad de procesamiento puede ser fácilmente reemplazada con un *cluster*. De este modo un *cluster* es considerado un supercomputador.

1.2.2 CARACTERÍSTICAS.

Un *cluster* debe cumplir con algunos requisitos o características que se detallan a continuación:

- Los nodos de un *cluster* están conectados entre sí por al menos un medio de comunicación. Los nodos que conforman el *cluster* deben estar interconectados entre sí a través de redes de alta velocidad; esto excluye a los sistemas de multiprocesamiento simétrico o SMP y sistemas de procesadores masivamente paralelos o MPP.

- Los *clusters* necesitan software de control especializado. Para tener un funcionamiento correcto de un *cluster*, este necesita un software de control especializado, el diseño y modelado del mismo depende del tipo de cluster utilizado. El software y las máquinas conforman el *cluster*.

- Software de control para gestión del cluster. Control que se refiere a la configuración del cluster, el cual depende del tipo de cluster y de la manera en que se conectan los nodos. El control puede ser de dos tipos según sea el caso:

    - Control centralizado. El cual consta de un nodo maestro o director, con el que se puede configurar todo el sistema.

    - Control descentralizado. Control en el que cada nodo se administra y gestiona individualmente.

- Homogeneidad de un *cluster*. Caracterizado por estar formado de nodos con arquitecturas y recursos similares. Este tipo de *cluster*s son implementados a nivel de sistema.

- Heterogeneidad de un *cluster. Cluster*s caracterizados por tener diferencias a nivel de tiempos de acceso, arquitecturas, sistemas operativos, estas dos últimas conllevan a tener bibliotecas de carácter general las cuales van a ser utilizadas como interfaz para poder formar un sistema conjunto entre los nodos. Este tipo de *cluster*s son implementados a nivel de aplicación.

1.2.3 ARQUITECTURA[9][10][11][12][13][14][20] [21]

La arquitectura de un *cluster* se muestra en la Figura 1-4.

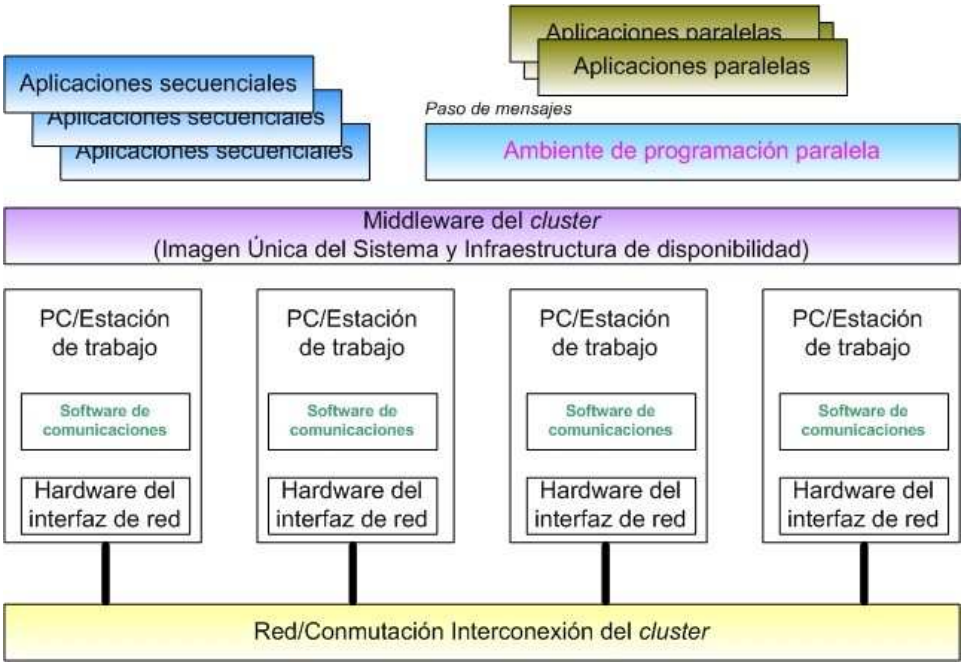


Figura 1-4. Arquitectura del *cluster*.

Como se puede observar la arquitectura posee varios componentes que se detallan en la siguiente sección.

1.2.3.1 Componentes de un *Cluster*

Los componentes de un *cluster* son: nodos de cómputo, sistemas operativos, redes de interconexión de alta velocidad o HSI (*high-speed interconnects*), *middleware*, entornos y herramientas de programación paralela y aplicaciones.

1.2.3.1.1 Nodos de Cómputo

Conformados por estaciones de trabajo, computadores personales, o SMPs (*Simetric Multiprocessors*), estos pueden estar interconectados a través de una red LAN (*Local Area Network)* o estar agrupados por una red SAN[7] (*Storage* o *System Area Network*).

Los nodos de cómputo están conformados a nivel de hardware por diferentes dispositivos los cuales permiten su funcionamiento individual y dentro del *cluster*: procesadores, memoria, caché, dispositivos de entrada/salida y buses de comunicación.

- Procesadores. Estos proveen la capacidad computacional del nodo. Pueden ser de diferentes fabricantes: Intel con sus familias Pentium 3, 4, Itanium y Xeon, HP y Compaq con su familia Alpha, dispositivos AMD (Athlon) entre otros.

- Memoria RAM (*Random Access Memory*). Es la memoria principal del computador la cual almacena los datos que está utilizando ese instante. Esta

[7] SAN (*Storage* o *System Area Network*) es una red de interconexión utilizada en un cuarto de computación dónde la distancia máxima del enlace es menor a 100 metros. Esta red es utilizada para conectar computadores a dispositivos de almacenamiento como arreglos de disco y también suele utilizarse para conectar computadores entre sí y conformar conglomerados o *clusters* de computadores personales. Redes de interconexión de alta velocidad actualmente permiten tener las dos funcionalidades de interconexión: almacenamiento y sistemas computacionales.

memoria se actualiza constantemente mientras el ordenador está en uso y pierde todos los datos cuando el sistema se apaga.

- Caché. La memoria caché es pequeña y de alta velocidad de acceso que sirve de búfer para aquellos datos de uso frecuente. La caché está conectada a la memoria principal la cual tiene una velocidad de acceso menor.

- Dispositivos de entrada/salida (I/O). Dispositivos de mucha utilidad para un computador, por medio de dispositivos de entrada puede escribirse en memoria desde el exterior, ubicándose junto con los datos sobre los cuales va operar, de igual manera debe existir dispositivos de salida para conocer los resultados de la ejecución de los programas. Los dispositivos de entrada/salida (I/O) permiten la comunicación entre el computador (memoria) y el mundo exterior (periféricos), dentro de dispositivos periféricos que pueden interactuar con un computador se tiene:

    - Dispositivos de presentación de datos. Dentro de este grupo están periféricos como el teclado, pantalla, ratón e impresora. Dispositivos que interactúan directamente con el usuario.

    - Dispositivos de almacenamiento de datos. Dispositivos que interactúan con la máquina, dentro de estos se encuentran las cintas magnéticas y discos magnéticos.

    - Dispositivos de comunicación con otros procesadores. Comunicación con otros procesadores remotos ya sea a través de una red LAN (*Local Area Networks*) , una red WAN (*Wide Area Networks*) o el bus de interconexión.

    - Bus del sistema. Periférico utilizado para transmisión de datos entre los diferentes dispositivos del computador con el procesador y la memoria.

1.2.3.1.2 Sistemas Operativos

Un nodo en un *cluster* (no siempre) es una entidad de cómputo autónoma, completa y que posee su propio sistema operativo. Los *clusters Beowulf* explotan las características sofisticadas de los sistemas operativos modernos para la administración de los recursos de los nodos y para la comunicación con los otros nodos a través de la red de interconexión.

Pueden utilizarse sistemas operativos modernos entre los que se tiene a Linux, Unix, Windows entre otros. Estos sistemas operativos son multitarea lo cual permite compartir recursos entre usuarios, dividir el trabajo entre procesos, asignar recursos cada proceso (memoria y ciclos de procesador) y tener al menos un hilo de ejecución.

1.2.3.1.3 Red de interconexión de altas velocidad

Existen diferentes redes de alta velocidad utilizadas en un *cluster*. Se puede citar algunas: Ethernet, ATM, SCI, cLAN, Myrinet, Infiniband y QsNet.

- Ethernet, Fast Ethernet y Gigabit Ethernet. Tecnologías muy utilizadas en el ámbito local, ethernet ofrece un ancho de banda de 10 Mbps, el cual en la actualidad es pequeño para soportar nuevas aplicaciones de gran demanda de acceso, es por ello que aparecieron las tecnologías fast ethernet, que brinda un ancho de banda de 100 Mbps y Gigabit Ethernet, que ofrece un ancho de banda de 1 Gbps.

- Modo de Transferencia Asíncrono (ATM – *Asyncronous Transfer Mode*).
  Tecnología desarrollada para hacer frente a la gran demanda de capacidad de transmisión para servicios y aplicaciones. Tecnología de alta velocidad basada en la conmutación de celdas o paquetes. ATM es una tecnología utilizada en ámbitos LAN y WAN, de vital importancia para aplicaciones en tiempo real como voz y vídeo, las cuales requieren garantía en la entrega de información, es decir necesitan de un protocolo orientado a conexión. ATM

es una de las redes de mayor ancho banda (soportada por Linux actualmente). La desventaja es que ATM no es barato, y todavía hay algunos problemas de compatibilidad al otro lado de distribuidores

- Interfaz Escalable Coherente (SCI – Scalable Coherent Interface). Es un estándar original de IEEE, es una tecnología de alto rendimiento y de bajo retardo, su funcionamiento se basa en la existencia de un anillo que contiene varios nodos conectados entre sí por enlaces unidireccionales punto a punto. Cada nodo que pertenece al anillo está conectado por un enlace proveniente del nodo que le precede y por un enlace con destino al nodo siguiente. Este enfoque se lo puede aplicar hasta un cierto límite de nodos, para ello existen diferentes arreglos los cuales pueden soportar hasta 64000 nodos configurados en diferentes topologías como anillos simples o múltiples, dependiendo éstos de las necesidades. La desventaja de esta configuración radica en que si un nodo falla, este incapacita a todos los nodos que comparten el anillo, también es imposible agregar o retirar un nodo del sistema debido a que su topología lo impide.

- cLAN. Tecnología creada por Gigante (hoy adquirida por Emulex [6]), ofrece un alto rendimiento para redes de alta velocidad, posee adaptadores PCI y switches de 8 y 30 puertos, ofreciendo velocidades de 1.25 Gbps por puerto (2.5 Gbps bidireccional). Es una tecnología muy utilizada en la interconexión de los diferentes dispositivos pertenecientes a un *cluster*.

- Myrinet. Tecnología de alta velocidad diseñada por Myricom [7] en Noviembre de 1998. El ancho de banda que maneja cada adaptador de red y los switch a incrementado desde 640 Mbps hasta los 2.4 Gbps, teniendo tiempos de entrega de paquetes que fluctúan entre los 7 y 10 micro segundos. Cada nodo posee una tarjeta de red PCI-X con una o dos conexiones, las cuales pueden manejar velocidades de 2 Gbps o 10 Gbps bidireccionales, estas tarjetas se interconectan a través de un cable Myrinet (fibra óptica) a un switch Myrinet de hasta 128 puertos. Esta tecnología posee un software el

cual detecta automáticamente a la red Myrinet sin necesidad de configurar el conmutador o *switch.*

- Infiniband. El estándar InfiniBand [8] es un bus de comunicaciones de alta velocidad, diseñado tanto para conexiones internas como externas. Para la comunicación utiliza un bus bidireccional con lo cual ofrece dos canales de transmisión independientes. El ancho de banda básico de un enlace simple (1x) es de 2.5 Gbits/s.

  Cada enlace simple puede tener 1, 4 ó 12 líneas de conexión, consiguiéndose unas velocidades de transferencia bidireccionales (permite una comunicación full duplex entre dispositivos) de 5 Gbits/s (1x), 20 Gbits/s (4x) y 60 Gbits/s (12x), respectivamente .

  Las redes Infiniband se basan en switch que interconectan los distintos dispositivos. De esta forma se tiene dos tipos de nodos: los intermedios (switch) y los finales (los distintos dispositivos de E/S, etc).

  Cada switch (o nodo intermedio) gestiona la comunicación entre los distintos subsistemas que interconecta, permitiendo múltiples envíos de datos simultáneos. Infiband a nivel de hardware posee dos tipos de conectores, HCA (*Host Channel Adapter*), interfaz utilizada para interconexión entre procesadores y TCA (*Target Chanel Adapter*), interfaz utilizada para conexión entre subsistemas de I/O. La Figura 1-5 muestra la arquitectura de esta tecnología.

- QsNet. Igual que las tecnologías myrinet e infiband, QsNet está conformada de dos partes, la interfaz de red Elan y el *switch* Elite, el cual dispone de 16 a 128 puertos, este *switch* posee dos canales virtuales bidireccionales por enlace, es decir cada enlace posee dos puertos de entrada/salida con una tasa de transferencia teórica de 400 Mbyte/s en cada dirección. El switch provee también dos niveles de prioridad, los cuales son de gran ayuda en la entrega de paquetes de mayor importancia en el menor tiempo posible.

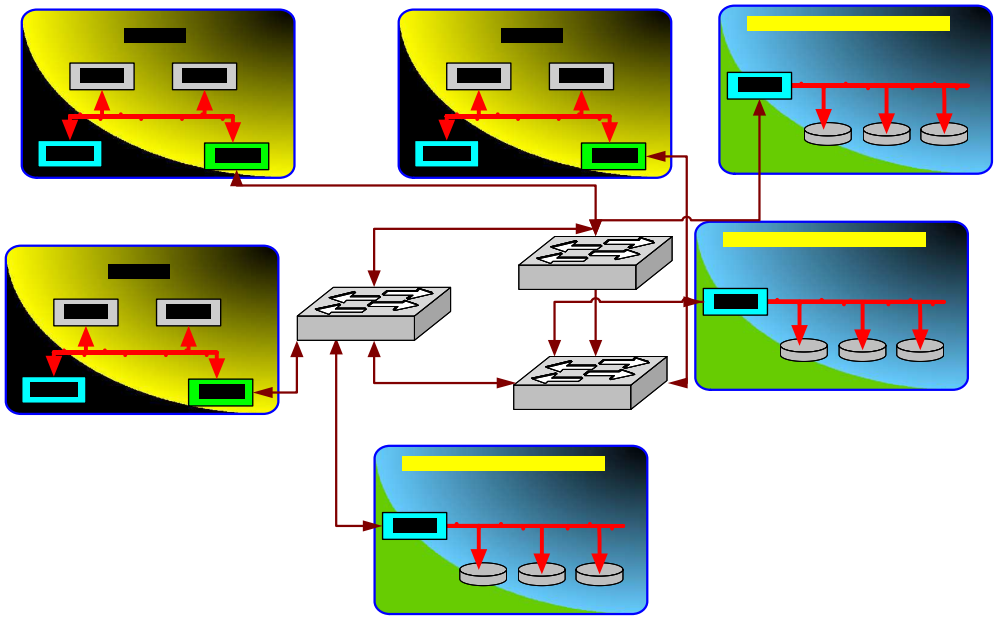


Figura 1-5.- Arquitectura de *Infiband*.

1.2.3.1.4 Middleware

Capa de software que se ejecuta sobre sistemas operativos y sistemas de comunicaciones heterogéneos ofreciendo una interfaz uniforme a las aplicaciones distribuidas. Funciona como una capa de abstracción de software distribuida que se sitúa entre las capas de aplicaciones y las capas inferiores: sistema operativo y red.

El *middleware* incluye sistemas de archivos con lo cual se consigue un sistema único de almacenamiento provisto por los discos de cada nodo, también provee diferentes ambientes de programación, sistemas de administración y planificación de tareas, así como da la oportunidad de trabajar con la Máquina Virtual de Java (JVM – *Java Virtual Machine*).

Middleware ofrece una Imagen Única del Sistema (SSI – *Single System Image*), la cual ofrece al usuario una visión unificada de todos los recursos del sistema, SSI presenta a los usuarios recursos disponibles y aplicaciones como un recurso único. SSI ofrece varios beneficios, dentro de los más importantes se tiene:

- Los nodos del *cluster* poseen una visión única de todos los recursos existentes.
- La administración y control del *cluster* se la puede hacer como una entidad única centralizada o descentralizada.
- El usuario no necesita conocer donde se ejecutan las aplicaciones.
- Los usuarios trabajan con interfaces familiares.
- El usuario puede conectarse al *cluster* como un sistema único, sin necesidad de hacerlo de manera individual a cada nodo como es el caso de un sistema distribuido.
- Presentar una completa transparencia al usuario de forma que este no tenga que preocuparse de detalles debajo nivel en la implementación, ni de cómo gestionar el sistema para optimizar su rendimiento.
- Escalabilidad del sistema, ya que los *clusters* pueden ampliarse fácilmente añadiendo nuevos nodos, las aplicaciones deben ser capaces de ejecutarse de forma eficiente en un amplio rango de tamaños de máquinas.
- Es importante la disponibilidad del sistema para soportar las aplicaciones de los usuarios, por medio de técnicas de tolerancia a fallos y recuperación automática sin afectar a las aplicaciones de los usuarios

1.2.3.1.5 Entornos y herramientas de programación paralela

- Hilos (*threads)*. Los hilos de ejecución son secuencias simples de instrucciones ejecutadas en paralelo con otras secuencias. Los hilos tienen la capacidad de dividir un programa en dos o más tareas que corren simultáneamente, por medio de la multiprogramación. En sistemas con multiprocesadores se utilizan los *threads* para usar de manera simultánea todos los procesadores disponibles. En sistemas con un procesador se los utiliza para optimizar el uso de los recursos del sistema.

  Un ejemplo de la utilización de hilos es tener un hilo atento a la interfaz gráfica (iconos, botones, ventanas), mientras otro hilo hace una larga

operación internamente. De esta manera el programa responde más ágilmente a la interacción con el usuario.

- Paso de Mensajes. Paso de Mensajes se implementa mediante librerías que permiten escribir programas paralelos para sistemas de memoria distribuida. Las implementaciones de estos sistemas consisten en un conjunto de librerías que proveen rutinas que pueden ser utilizadas en programas escritos en los lenguajes de programación C, C++ y Fortran. Existen diversos sistemas de paso de mensajes, donde los más usados son la Máquina Virtual Paralela (PVM – *Parallel Virtual Machine*) y el Interfaz de Paso de Mensajes (MPI – *Message Passing Interface*).

- Sistemas de Memoria Compartida Distribuida (DSM). Sistema mediante el cual cada procesador posee su memoria local y se interconecta con otros procesadores por medio de un dispositivo de alta velocidad, y todos ven las memorias de cada uno como un espacio de direcciones globales. Es un sistema escalable y fácil de programar. La Figura 1-6 muestra la arquitectura de este modelo de programación.

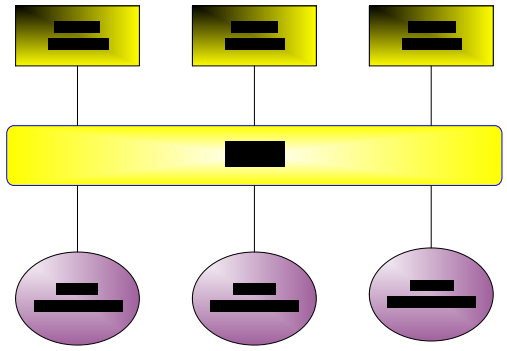


Figura 1-6 .- Arquitectura DSM.

- Depuradores Paralelos. Herramientas muy útiles que permiten desarrollar aplicaciones de alto rendimiento, correctas y eficientes. Un depurador paralelo ofrece las siguientes utilidades:

- Administrar múltiples procesos y threads dentro de un proceso.
- Mostrar código fuente.
- Permitir mostrar objetos, subrutinas y funciones.
- Utilizar puntos de parada en código fuente.
- Mostrar arreglos y sus elementos.
- Manipular variables y constantes.

- Herramientas de Análisis de Rendimiento. Herramientas que facilitan el control y funcionamiento de una aplicación ya que permiten analizar y localizar puntos de fallo.

- Herramientas de Administración. La administración de un *cluster* es de vital importancia para mantener el correcto funcionamiento de éste, mediante estas herramientas se puede tener un monitoreo adecuado de sus componentes.

1.2.3.1.6 Aplicaciones

Estas pueden ser paralelas o distribuidas y secuenciales.

- Aplicaciones Paralelas. Son aplicaciones que se distribuyen entre varias máquinas y plataformas para trabajar en forma integrada, típicamente sobre una red, para realizar una variedad de funciones relacionadas, tal es el caso de la arquitectura cliente/servidor caracterizado por dividir la funcionalidad de la aplicación en dos papeles: cliente y servidor. El servidor se encarga de proporcionar una serie de servicios al cliente, los cuales son utilizados por los clientes para completar la funcionalidad de una determinada aplicación.

- Aplicaciones Secuenciales. Es aquella en la que una acción (instrucción) sigue a otra en secuencia. Las tareas se suceden de tal modo que la salida de una es la entrada de la siguiente y así sucesivamente hasta el fin del proceso.

1.2.4 CLASIFICACIÓN.[15][16][17][18]

Los *clusters* dependiendo de su aplicabilidad pueden clasificarse de diferentes maneras. La clasificación más generalizada es la que se presenta a continuación:

- Alto rendimiento (HP, high performance) .
- Alta disponibilidad (HA, high availability).
- Balanceo de Carga (Load Balancing).
- Alta Confiabilidad (HR, high reliability).

1.2.4.1 Alto rendimiento (HP, *high performance*).

Este tipo de *cluster* lo que busca es suplir las necesidades de súper computación para resolver problemas de determinadas aplicaciones que requieren un alto procesamiento, esto se logra mediante la utilización de un grupo de máquinas individuales las cuales son interconectadas entre sí a través de redes de alta velocidad y de esta manera se obtiene un sistema de gran rendimiento que actúa como uno solo.

La utilidad principal de este tipo de *cluster* es principalmente en aplicaciones en las que se requieren gran capacidad de procesamiento computacional, la cual soluciona problemas de alto procesamiento mediante la utilización de técnicas necesarias para la paralelización de la aplicación, distribución de los datos a los nodos, la obtención y presentación de resultados finales.

Generalmente estos problemas de cómputo suelen estar ligados a:

- Cálculos matemáticos.
  - Estado del tiempo.
  - Cifrado y Descifrado de códigos.
  - Compresión de datos.
  - Astronomía.
  - Simulación Militar.

1.2.4.2 Alta disponibilidad (HA, *high availability*).

Cluster muy solicitado y de mucha importancia para empresas que brindan servicios 24-7 dónde su principal función es la de mejorar los servicios que dichas empresas ofrecen a los clientes en las redes a las que pertenecen, sean estas internas (*intranet*) o externas (Internet).

El brindar alta disponibilidad no hace referencia a conseguir una gran capacidad de cálculo, si no lograr que una colección de máquinas funcionen en conjunto y que todas realicen la misma función que se les encomendó. La característica principal de este *cluster* es que ante la existencia de algún problema o fallo de uno de los nodos, el resto asumen ese fallo y con ello las tareas del nodo con problemas. Estos mecanismos de alta disponibilidad lo brindan de forma transparente y rápida para el usuario.

La escalabilidad en un *cluster* de alta disponibilidad se traduce en redundancia lo cual garantiza una pronta recuperación ante cualquier fallo.

La flexibilidad y robustez que poseen este tipo de *cluster* los hacen necesario en sistemas cuya funcionalidad principal es el intercambio masivo de información y el almacenamiento de datos sensibles, dónde se requiere que el servicio esté presente sin interrupciones.

El mantenimiento es otra de las ventajas que ofrece. El mantenimiento se puede realizar de manera individual a cada máquina que compone el conglomerado evitando comprometer los servicios que este brinda.

Existen dos tipos de configuraciones aplicables a estos *clusters*:

- Configuración activo – pasivo: Esta configuración tiene dos actividades en los nodos que componen el *cluster*, los activos son aquellos que se encargan de ejecutar las aplicaciones encomendadas, mientras que los nodos restantes actúan como respaldos redundantes para los servicios ofrecidos.

- Configuración activo - activo: En este caso, todos los nodos actúan como servidores activos de una o más aplicaciones y potencialmente como respaldos para las aplicaciones que se ejecutan en otros nodos. Cuando un nodo falla las aplicaciones que se ejecutaba en él migran a uno de los nodos de respaldo.

1.2.4.3 Balanceo de carga.

Técnica muy utilizada para lograr que un conjunto de servidores de red compartan la carga de trabajo y con ello el tráfico de sus clientes. Este proceso de dividir la carga de trabajo entre los servidores reales permite obtener un mejor tiempo de acceso a las aplicaciones y con ellos tener una mejor confiabilidad del sistema. Además como es un conjunto de servidores el que atiende el trabajo, la falla de uno de ellos no ocasiona una falla total del sistema ya que las funciones de uno, las puede suplir el resto.

1.2.4.4 Alta Confiabilidad (HR, *high reliability*).

Cluster caracterizado por ofrecer una alta confiabilidad al sistema. La idea es obtener respuestas eficientes del sistema a pesar de tener una sobrecarga de las capacidades de un servidor. Estos *clusters* se caracterizan por ejecutar un mayor número de tareas en el menor tiempo posible.

1.2.4.5 *Cluster*: Funcionamiento.

El funcionamiento de un *cluster* está basado en dos partes fundamentales: a nivel de software y a nivel de hardware.

- A nivel de Software. Mediante el uso de un sistema operativo que brinde las herramientas necesarias para la creación de un *cluster* , tal es el caso de un kernel Linux modificado, compiladores y aplicaciones especiales, los cuales permitan que los programas que se ejecutan en el sistema exploten todas las ventajas del *cluster*.

- A nivel de Hardware. Mediante la interconexión entre máquinas (nodos) del *cluster*, las cuales se juntan utilizando redes dedicadas de alta velocidad como por ejemplo Gigabit Ethernet.

Cuando se trata de brindar balanceo de carga mediante un *cluster* el hardware y software trabajan conjuntamente para distribuir la carga de tráfico a los nodos, para de esta manera poder atender eficientemente las subtareas encomendadas y con ello la tarea general asignada al *cluster.*

Un servicio de alta disponibilidad en el *cluster* normalmente no distribuye la carga de tráfico a los nodos (balanceo de carga) ni comparte la carga de procesamiento (alto rendimiento) sino más bien su función es la de estar preparado para entrar inmediatamente en funcionamiento en caso de que falle algún otro servidor..

1.2.4.6 Características y funcionamiento de alta disponibilidad.

1.2.4.6.1 Sistemas de alta disponibilidad y sistemas tolerantes a fallos.

Los sistemas tolerantes a fallos son aquellos sistemas que solucionan problemas de fallo del hardware de algún dispositivo, la tolerancia a fallos hace que un sistema pueda detectar y actuar instantáneamente para reestablecer el servicio brindado.

Los sistemas de alta disponibilidad como ya se vio en el apartado anterior, involucra el tener servidores que actúan entre ellos como respaldos vivos de la información que sirven. Este tipo de *cluster*s se les conoce también como *cluster* de redundancia.

En los últimos años la idea de alta disponibilidad y de tolerancia a fallos se ha ido acercando, con el surgimiento de nuevas tecnologías y el abaratamiento del hardware se ha logrado que con la idea de alta disponibilidad se logre tener un sistema tolerante a fallos a bajo precio.

1.2.4.6.2 SPOF (Single Point of Failure ó Punto Simple de Fallo).

SPOF hace referencia a la tenencia de un elemento no replicado que puede estar sujeto a fallos, logrando con esto la suspensión del servicio que se está brindando. Es por ello la importancia de evitar tener un SPOF en los subsistemas del sistema general, ya que con ello se pondría en peligro la prestación continua de servicios del sistema.

En sistemas de alta disponibilidad a mas de tener redundancia en sus servidores, es importante tenerla en otros dispositivos que componen el *cluster*, tal es el caso de dispositivos de interconexión, red de comunicación de servidores; etc. Esto con la finalidad de evitar el tener un SPOF a nivel de subsistemas y sistemas como tal que conforman el *cluster* que se está implementando.

1.2.4.6.3 Servicio de Datos.

El servicio de datos hace referencia al servicio y sus respectivos recursos que se esté brindando a un cliente. En entornos de alta disponibilidad al servicio brindado y al grupo de recursos se denominan logical host o software package.

Los recursos que se estén utilizando deben tener mecanismos necesarios que permitan la suplantación y conmutación física entre los nodos cuando uno de estos falle, logrando de esta manera que el servicio de datos ofrecido falle en su funcionamiento. De esta manera la única afección que el sistema tendrá es en el tiempo de conmutación en la puesta en marcha del servicio de datos.

1.2.4.6.4 Dinámica de Alta Disponibilidad.

Dinámica que hace referencia a las reconfiguraciones que el *cluster* debe hacer para garantizar la máxima disponibilidad de un determinado sistema; va orientada a los nodos que conforman el *cluster* y la forma de cómo éste responde. Existen diferentes maneras de cómo el sistema responde ante la presencia de un fallo, entre las cuales se tiene:

- Tolerancia a fallos (*failover*). Se da cuando un nodo falla y otro debe asumir sus responsabilidades. Para ello el nodo entrante debe importar los recursos del nodo con fallo y habilitar los servicios de datos.

- Toma de control o tolerancia a fallos automático (*takeover*). Se produce cuando el servicio de datos falla y se detecta por un determinado nodo, a este nodo se lo considera nodo fallido y se ve forzado a ceder sus servicios y recursos. En este caso se requiere una monitorización continua del servicio de datos.
- Tolerancia a fallos manual (*switchover* o *giveaway*). Se caracteriza por ceder los recursos y servicios de datos de un nodo a otro. Se utiliza cuando se realizan tareas de mantenimiento y administración a un nodo.

- División de cerebros (*splitbrain*). Mecanismo utilizado cuando el proceso de gestión de un cluster HA[8] falla. Esta falla se da debido a problemas en la comunicación y verificación de los nodos existentes. Debido a que los nodos no conocen de la existencia de sus vecinos y asumen que son los únicos en el sistema, por tal motivo cada nodo intentará apropiarse de todos los recursos del sistema incluyendo el servicio de datos y tener el control total del cluster. El *splitbrain* es un caso especial del *fileover*.

  El *splitbrain* puede dejar a un sistema fuera de funcionamiento. Se puede evitar utilizando dos métodos. El primero es actuar de forma prudente ante esta falla, utilizando los recursos compartidos como señal de estar activos, luego de que un nodo constata el problema debe reservar un recurso compartido llamado *quórum*, este recurso debe ser reservado por un solo nodo y el nodo que llegue tarde a la reserva entiende que debe abandonar el *cluster* y ceder todos sus recursos. El recurso *quórum* únicamente es utilizado como método de decisión para abandonar o no un *cluster*. El segundo método consiste en tratar de dejar fuera o apagar al nodo con fallo una vez detectado

[8] HA: *High Availavility*

el problema, el primer nodo que lo haga toma el control del *cluster* y con ello el control de todos los recursos.

1.2.4.6.5 Recursos de un Servicio de Datos.

Al trabajar con un *cluster* de alta disponibilidad se tienen diferentes tipos de recursos que son fundamentales para su funcionamiento y que se listan a continuación:

- Recursos Computacionales. Son aquellos que permiten alojar y ejecutar el servicio de datos brindado. Estos recursos se consideran a nivel de CPU y el *cluster*. En alta disponibilidad se tienen recursos a nivel de *cluster* dónde cada nodo que lo conforma debe tener una copia en memoria del programa de servicio de datos.

- Recursos de Comunicaciones. Recursos utilizados para brindar el acceso al servicio de datos mediante una red de comunicaciones.

- Recursos de Almacenamiento. Son los recursos más críticos en alta disponibilidad. Se debe garantizar la integridad y confiabilidad de los datos almacenados en los discos de los nodos o servidores. La falla de estos dispositivos hace que los datos se corrompan y con ello se tenga efectos irreversibles que afecten el rendimiento del sistema.

1.2.4.7 Balanceo de carga (*Load balancing*).

El balanceo de carga permite suplir las necesidades del inminente crecimiento del tráfico en Internet. Existen dos alternativas para manipular el crecimiento del tráfico de Internet: la primera permite usar una máquina de grandes características y de alto precio que probablemente a futuro quede obsoleta; la segunda alternativa consiste en utilizar un conjunto de servidores virtuales o granja de

servidores de bajo costo que trabajan en conjunto para balancear la carga entre ellos.

El balanceo de carga consiste en compartir la carga de trabajo y tráfico de los clientes que éstos acceden. Al grupo de servidores que prestan este servicio se los conoce como servidores virtuales.

Al balancear la carga se mejora el tiempo de respuesta, acceso y confiabilidad. La caída de un servidor no influye en el funcionamiento de todo el *cluster* ya que las funciones de éste son asumidas por el resto de servidores virtuales. Cuando la carga de trabajo sea mayor se pueden añadir más servidores al *cluster* y escalar este sistema para garantizar el balanceo de carga.

Existen diferentes métodos de distribución de carga que se detallan a continuación.

1.2.4.7.1 Balanceo de Carga por Sistema de Nombres de Dominio o DNS.

Este proceso consiste en crear un dominio DNS[9] al cual se le asigna diferentes direcciones IP[10] pertenecientes a los servidores que están funcionando. A esta configuración no se le considera un *cluster* debido a que en la caché del explorador de Internet de los clientes se almacena la dirección IP del servidor que en ese momento lo atendió. Los clientes automáticamente redireccionan las peticiones refiriéndose a la dirección del servidor de la caché cuando un nuevo

---

[9] Sistema de Nombres de Dominio o DNS (*Domain Name System*): El sistema de nombres de dominio es un estándar de Internet. Este se describe en el RFC (*Request For Comments*) 1034 y 1035. El propósito de DNS es crear un sistema que permita realizar búsquedas en una base de datos tipo árbol. Estas búsquedas se realizan para la obtención de la dirección IP o del nombre de *host* que pertenece a un nodo que se encuentra en el sistema de nombres de dominio.

[10] IP: *Internet Protocol*, protocolo de comunicaciones de la capa red de la arquitectura TCP/IP Es un protocolo no orientado a conexión, no confiable; es decir no garantiza la comunicación entre los extremos.

requerimiento se produce. El balanceo de carga no se realiza por completo ya que un servidor puede atender solicitudes de gran procesamiento, mientras otros servidores pueden atender solicitudes más sencillas, esto se debe a que carecen de un dispositivo que controle y redireccione equitativamente el trabajo a los servidores, la falta de este dispositivo hace que la repartición de carga no sea equitativa y ello conlleva a tener un desigual funcionamiento de los servidores.

Cuando un servidor cae, los clientes que eran atendidos por este van a continuar enviando requerimientos sin que sus peticiones sean cumplidas, quedando sin atención hasta el momento de que su caché se actualice y aprenda la dirección de un servidor operativo.

1.2.4.7.2 Balanceo de Carga mediante un Nodo Director.

Mecanismo basado en la utilización de un nodo director. El nodo director recibe todas las peticiones de los clientes, balancea la carga y redirecciona a los servidores del *cluster.*

El nodo director reenruta las peticiones a los nodos que posean menor carga para que atiendan el requerimiento solicitado, una vez procesada la petición los servidores virtuales devuelven el resultado al nodo director para que este entregue los resultados al cliente. Existen variaciones donde los nodos o servidores virtuales pueden entregar el resultado al cliente directamente.

Existen tres tipos de reenvío de paquetes que el nodo director hace los servidores que conforman el *cluster*:

- Balanceo por NAT (*Network Address Traslation*). Mecanismo basado en trasladar las direcciones IP origen/destino de los paquetes que llegan al nodo director y los reenvía a uno de los servidores reales disponibles. Luego de que los servidores internos procesan el paquete recibido lo reenvían al nodo

director. El único mecanismo para que todos los servidores del *cluster* puedan salir al Internet es a través del nodo director.

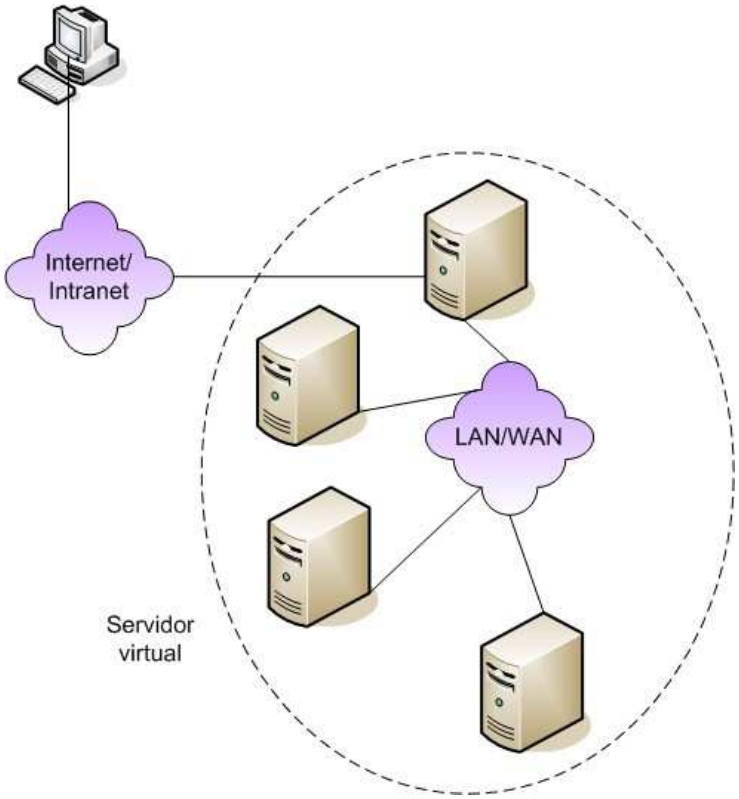


Figura 1-7. Balanceo de carga mediante un nodo director.

- Balanceo por encapsulado IP. Mecanismo basado en el encapsulamiento IP (IP *tunneling*). La unidad de datos TCP/IP que llega al nodo director es encapsulada dentro de otra unidad de datos manteniendo las direcciones origen/destino intactas. Esta nueva unidad de datos contiene los datos originales del cliente que genera la petición, posee la dirección origen del nodo director y la destino del servidor disponible para atender el requerimiento

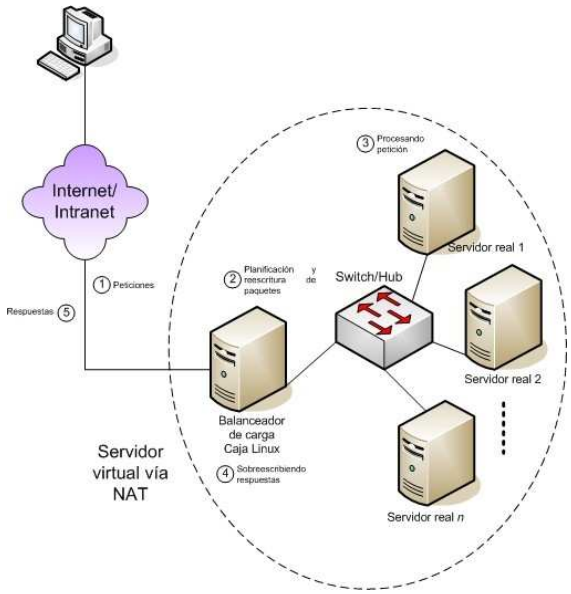


Figura 1-8. Balanceo de carga mediante NAT.

El servidor que atiende el requerimiento desencapsula la unidad de datos que le envió el nodo director. Los servidores virtuales tienen configurada en una de sus interfaces de red la misma dirección pública del nodo director con la finalidad de poder aceptar la unidad de datos original y servir la petición requerida.

Figura 1-9. Transmisión y recepción de unidad de datos mediante balanceo por encapsulación.

Luego de procesar las peticiones se las reenvía al cliente directamente utilizando la dirección pública del *cluster*, y así evitar reenrutar el resultado al nodo director para que este la entregue al cliente que originó la petición. Así se logra evitar un cuello de botella en el nodo director, haciendo más fácil el proceso de atención de peticiones.

La distribución de los servidores internos del *cluster* se lo puede realizar a nivel de una red WAN sin necesidad de tenerlos en un solo segmento de red logrando con ello evitar tener un punto de fallo único en caso de que el segmento quede fuera.

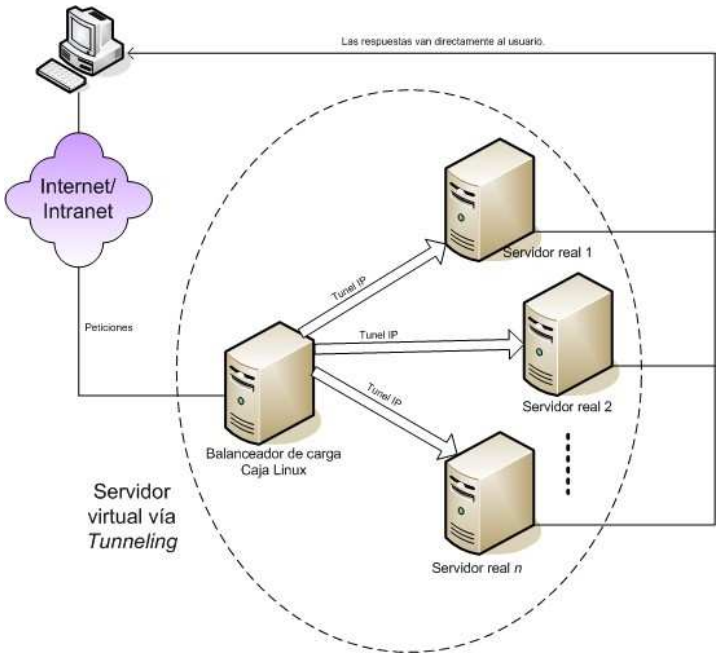


Figura 1-10. Balanceo de carga por medio de encapsulado.

- Balanceado por enrutamiento directo. Todos los servidores del *cluster* incluido el nodo director están en un mismo segmento de red físico; a su vez todos los nodos comparten la misma dirección IP pública. En este mecanismo se evita sobrecargar al nodo director con traslación de direcciones IP y evita el realizar encapsulamiento de la unidad de datos recibida. De este modo se evita tener un cuello de botella en el nodo director. Las respuestas del servidor quien atendió el requerimiento se envían de forma directa a los clientes.

  La configuración de este sistema debe garantizar que los nodos no respondan a comandos ARP[11] para evitar conflictos con otros protocolos,

[11] ARP (*Address Resolution Protocol*): protocolo de capa enlace del modelo ISO/OSI para determinar la dirección MAC de los partícipes de una comunicación.

logrando con esto que todos los servidores respondan con la misma IP pero con diferentes direcciones MAC[12].

Cuando llega una petición al nodo director decide a que servidor interno enviar dicha petición, este redireccionamiento se realiza a nivel de capa enlace utilizando la dirección MAC del servidor que va atender el requerimiento. Cuando el paquete llega a éste servidor se analiza hasta el nivel de red IP. Debido a que posee la dirección IP pública del *cluster* acepta el paquete y lo procesa, luego de esto reenvía el resultado al cliente.

Existen diferentes algoritmos que permiten distribuir la carga en un *cluster*.

- *Round Robin*. También llamado FIFO (*First In First Out*), proceso en el que el nodo director envía una petición a un determinado servidor del *cluster,* la siguiente petición se envía al siguiente servidor disponible y así sucesivamente hasta llegar al último servidor disponible, luego del cual se vuelve a enviar al primero.

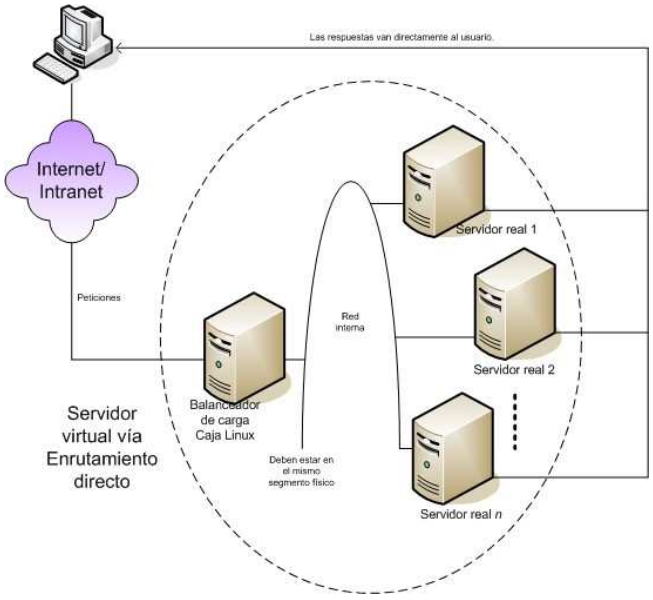


Figura 1-11. Balanceo de carga por enrutamiento directo.

[12] MAC: (*Medium Access Control*) Subcapa de la capa de enlace del modelo ISO/OSI perteneciente al estándar IEEE 802.3. Permite establecer la comunicación a nivel físico entre interfaces a través de una dirección única de 48 bits.

- *Round Robin* ponderado. Su funcionamiento es igual al *Round Robin*, solo que a cada servidor se le da un indicativo de acuerdo a su capacidad de procesamiento, es decir servidores con alta capacidad de procesamiento recibirán mayor carga que los que poseen una menor capacidad.

- Servidor con menos conexiones activas[13]. Proceso caracterizado por tener un mecanismo continuo de consulta que permite verificar cuántas conexiones activas poseen los servidores que están atendiendo los requerimientos del cliente. Dependiendo del resultado de esta consulta el nodo director direcciona las peticiones a los servidores con menos carga. Este proceso es muy útil cuando todos lo servidores poseen una capacidad de procesamiento similar (sistema homogéneo), logrando tener de esta manera un trabajo equilibrado.

- Servidor con menos conexiones activas ponderado. Este algoritmo consiste en asignar una mayor carga a los servidores con menos conexiones activas pero verificando la capacidad de procesamiento de éstos y dependiendo de esta capacidad asignar una mayor o menor carga a los servidores. Se pondera a los servidores por su capacidad de procesamiento y ese valor se utiliza para la asignación de carga.

- Menos conectado basado en servicio. Algoritmo caracterizado por dirigir todas las peticiones a un mismo servidor hasta lograr que se sobrecargue, es decir que tenga un gran número de conexiones activas que superen a su capacidad de procesamiento, luego de esto se ejecuta el algoritmo de menos conexiones activas ponderadas sobre el resto de servidores del *cluster* y se pasan las peticiones al servidor con menos conexiones activas analizando el valor ponderado de este; el proceso continua hasta que se

[13] Conexiones activas: Son conexiones establecidas entre el cliente y el nodo director.

satura la capacidad de conexiones activas. Este mecanismo se mantiene de manera sucesiva hasta llegar al último servidor en servicio del *cluster.* Este algoritmo es muy utilizado cuando se ofrece varios servicios distintos y se quiere especializar a una determinada máquina en un servicio determinado. Todas las máquinas son capaces de reemplazar a cualquiera de lo servidores activos cuando uno de estos falle.

- Tablas *hash*[14] por origen y destino. Proceso que se basa en la utilización de una tabla de asignaciones fijas la cual contiene las direcciones IP origen y destino, en base a estas se indica qué servidor deberá atender la petición. El nodo director o balanceador compara las direcciones de las tramas TCP/IP que reciba con estas tablas y actúa en consecuencia.

- Conexiones Persistentes. Mecanismo útil ya que a los algoritmos anteriormente analizados se les puede añadir la condición de que una vez que un cliente se ha conectado con un servidor, siempre las peticiones se le dirija al mismo servidor.

1.3 PLANIFICACIÓN Y ADMINISTRACIÓN DE *CLUSTERS* [18]

1.3.1 ADMINISTRACIÓN

Luego del proceso de instalación del hardware, configuración del *cluster*, se necesita que estos recursos sean administrados para tener un control de rendimiento y funcionamiento del *cluster*.

[14] Tablas hash: Una tabla hash es una estructura de datos que permite el acceso de forma directa a los elementos que almacena a partir de una clave generada como un resumen (hash) de parte de los propios datos.

La administración de un *cluster* es de mucha importancia en el funcionamiento del mismo, ya que es fundamental administrar adecuadamente los recursos del *cluster* y con ello poder detectar fallos a nivel de software y hardware, además poder monitorear el rendimiento de cada servidor con la finalidad de constatar su correcto funcionamiento y en caso de tener algún inconveniente poder tomar mediadas preventivas y evitar que estos colapsen.

Para poder lograr una administración adecuada de un *cluster* se debe considerar los siguientes aspectos:

- Registro de eventos (logging)
- Falla y Recuperación de Hardware.
- Falla de Software.
- Falla y Recuperación del Sistema de archivos.
- Encolamiento (*Queing)*
- Planificación (Scheduling)
- Monitoreo (Monitoring)
- Contabilidad (Accounting)

Existe una relación entre usuarios, recursos y las actividades involucradas en la administración del *cluster*. El software de administración se ubica entre los usuarios y los recursos (servidores reales y director) del *cluster*. En primera instancia los usuarios envían sus peticiones a una cola que almacena los trabajos que se van a llevar a cabo (el usuario puede en cualquier momento pedir información sobre el estado de ese trabajo). Los trabajos esperan en la cola hasta el momento en que ingresan al *cluster* para ser atendidos. Mientras todo está ocurriendo, el sistema de administración esta monitoreando el estado de los recursos del sistema y cuáles usuarios los están utilizando.

A continuación se va analizar con más detalle las actividades que se deben llevar a cabo en un sistema de administración:

1.3.1.1 Registro de eventos (logging)

El registro de eventos es el proceso por el cual todos los aspectos relacionados al funcionamiento de una máquina y operación del *cluster* se guardan en un archivo de registro (*logs*) el cual puede ser usado para futuras consultas.

El administrador de un sistema Linux tiene a su disposición un programa de generación de registros que monitoriza todos los eventos que se producen en el mismo. Syslog guarda, analiza y procesa todos los archivos de registro sin requerir apenas intervención por parte del administrador. La ubicación del directorio donde se almacenarán los registros depende del servicio instalado, por lo general el directorio donde se almacena es el /var/log.

1.3.1.2 Falla y Recuperación de Hardware.-

Una de las principales responsabilidades de la administración de un *cluster* es la falla a nivel de hardware. El impacto que ocasiona la falla del hardware puede conllevar a tener todo el sistema inoperante. Estos impactos pueden ocasionar pérdidas de los servidores de archivos, servidores de red, dispositivos de ínterconectividad, entre otros, siendo importante la necesidad de tener monitoreado el funcionamiento de estos y a la vez tomar medidas de prevención.

Se debe tener presente que muchos fallos a nivel de hardware que se presentan no afectan mucho al rendimiento general del *cluster*, como es el caso de la falla del disco de un servidor, éste dejaría de operar y afectaría únicamente a los cliente que a él se conectan, pero en términos generales no afectarían a todo el sistema ya que el *cluster* de por si se encarga de suplir ese fallo, pero a fin de cuentas esa falla si afecta al rendimiento del sistema, en especial al tiempo de respuesta de los servidores operativos en el *cluster.*

Para evitar un gran impacto de estos inconvenientes y poder tener una recuperación ante fallos del hardware, la administración debe considerar varios

aspectos como el aislar el componente fallido con la finalidad de que puedan interferir con el funcionamiento del resto de componentes del *cluster*, tener respaldos (backups) a nivel de hardware para poder suplantar de manera rápida el componente fallido y así poder garantizar la disponibilidad del sistema.

1.3.1.3 Falla de Software.

La falla ocasionada por el software al igual que los fallos a nivel de hardware son críticos debido a que pueden dejar inoperante al *cluster*, pero a la vez estas fallas involucran otros aspectos que deben ser tomados en cuenta. Las fallas de software muchas veces tienen arreglo, otras veces no, pero es de vital importancia detectarlas, para poder analizar las posibilidades de evitar que vuelvan a ocurrir, por lo general el Linux los errores se los evita con los denominados parches del sistema, los cuales vienen a suplir las fallas del sistema a nivel de software.

Para que el sistema tenga un funcionamiento adecuado y a su vez poder evitar los problemas de indisponibilidad que mencionada falla acarrea, se debe tomar en consideración varios aspectos tales como: la actualización del sistema debe hacérsela de manera continua, una vez que algún parche actualizado haya sido desarrollado. Realizar pruebas de funcionamiento del sistema posterior a la actualización del mismo, para poder de esta manera constatar las debilidades que mencionada actualización trajo al sistema y si no las resuelve, tener la posibilidad de volver a las versiones de software anteriores.

1.3.1.4 Falla y Recuperación del Sistema de archivos.-

La falla en un sistema de archivos es muy crítica, se podría decir que es la pérdida que mayor daño ocasionaría a un sistema, ya que se perdería toda la información almacenada por varios meses o años, en especial se perdería los datos de usuario y de las aplicaciones que hacen uso de éste.

Por ello es de vital importancia tomar medidas preventivas que logren evitar tener un punto de fallo en el sistema de archivos, teniendo sistemas de archivos redundantes mediante el uso de técnicas como RAID, poseer un sistema de archivos con *journaling* el cual permita recuperar rápidamente los datos en caso de existir alguna falla inesperada en el sistema, poseer un sistema de archivos paralelo, con lo cual la perdida de un servidor de archivos no influya de manera total al funcionamiento general del sistema ya que la existencia de otros servidores de archivos podrían suplir su ausencia.

1.3.1.5 Encolamiento *(Queing).-*

Un aspecto importante a tomar en cuenta en la administración de un *cluster* es el encolamiento (Q*ueing)* que consiste en el proceso de acumular los trabajos para que sean ejecutados por el conjunto de recursos disponibles. Las tareas y los trabajos que los usuarios desean realizar, son presentados por el sistema de administración como un conjunto llamado grupo de trabajo.

Este grupo de trabajo para ser ejecutado necesita de dos aspectos fundamentales que el sistema debe proveer, en primer lugar proveer los recursos necesario (como la cantidad de memoria o CPU's necesitados), y en segundo lugar una descripción de las tareas a ser ejecutadas (archivo a ejecutar, datos requeridos para procesar cualquier petición; entro otros).

El grupo de trabajo luego de presentado al sistema de administración, es colocado en una cola hasta que el sistema provea los recursos necesarios (por ejemplo: la cantidad correcta de memoria y CPU requerido) para procesar cualquier trabajo encomendado. El tiempo de espera para que los trabajos se ejecuten depende exclusivamente del tiempo de ocupación de los recursos.

1.3.1.6 Monitoreo (Monitoring).

El monitoreo involucra el observar que el rendimiento y funcionamiento del *cluster* sea correcto. Una operación correcta implica tener a todos los recursos de

hardware y software monitoreados y con ello constatar que el funcionamiento sea el esperado.

El tener monitorizado un sistema es muy importante en el proceso de administración del *cluster* ya que mediante los datos que arroje dicho monitoreo se puede prever cual es el comportamiento de un determinado dispositivo que compone el *cluster*, y con ello poder tomar los correctivos necesarios para evitar una caída del sistema.

Es indispensable chequear parámetros de los equipos cuando estén en funcionamiento, parámetros como: arquitectura y frecuencia del CPU, tipo y versión del sistema operativo, memoria total, número de CPU's en cada servidor, tiempo de actividad, porcentajes de uso de CPU, utilización de disco, memoria disponible, actividad de procesos, entro otros; y de esta manera constatar su correcta operación.

1.3.1.7 Contabilidad (Accounting).

Mecanismo utilizado para la recolección de datos de cada grupo de trabajo que se ejecuta en el *cluster*, el *accounting* es una herramienta muy importante que permite generar información útil para el proceso de planificación de tareas, esta herramienta puede ser utilizada para varios propósitos tales como:

- Elaboración semanal de reportes de utilización del sistema.
- Elaboración de reportes de utilización mensual de recursos por usuario.
- Elaboración de un cronograma de políticas de planeamiento.
- Prever requerimientos computacionales futuros.
- Determinar áreas que deben ser mejoradas dentro del sistema.

1.3.2 PLANIFICACIÓN DE TAREAS

La planificación de tareas es una actividad de mucha utilidad en la administración de un cluster, ya que permite programar un conjunto de actividades que se

ejecutarán de acuerdo al tiempo, necesidad y circunstancia que un determinado sistema así lo requiera. Al hablar de tarea se hace referencia a un programa ejecutable que realiza determinadas funciones. Un proceso consiste en un número de tareas con cierta independencia que se coordinan para cumplir funciones lógicas.

Un planificador de tareas debe garantizar tres aspectos fundamentales:

- Rendimiento. Optimizar la ejecución del número de tareas y procesos.
- Tiempo de respuesta. Conseguir que el tiempo de ejecución de un determinado proceso sea mínimo.
- Optimización de CPU. Optimización constante de la carga de proceso de la CPU.

Este tipo de herramientas permiten la ejecución automática de tareas, esto mediante la ejecución de comandos los cuales se ejecutarán dependiendo del tiempo y las necesidades que se presenten, en Linux hay tres formas de especificar tiempos y comandos:

- at. el comando se inicia en un tiempo determinado (por ejemplo: at [-f file] time [date])
- batch. el comando se inicia una vez como la carga de una función del sistema (por ejemplo: batch [archivo])
- crontab. al igual que at especifica el tiempo al cual se ejecutará un programa script.

En el caso de la planificación de tareas multiplataforma (desde Linux pasando por Unix o Windows), el principal objetivo es gestionar tantas tareas como sea posible de la manera más rápida posible y con el menor número de anomalías.

Una solución de planificación de tareas debe ser capaz de interrumpir, parar o reiniciar tareas, así como el poder asignar prioridades a la ejecución de las

mismas. También es importante la existencia de puntos de verificación, los cuales van a proporcionar una imagen del estado actual de la tarea en ejecución, de igual manera la tarea puede continuar su ejecución desde el punto de verificación.

Los planificadores de tareas completos y modernos pueden combinar múltiples tareas en grupos cuando lo necesitan y procesarlo como una única unidad cuyo resultado será utilizado como condición de inicio para otras tareas o grupos de tareas.

Bibliografia Capítulo 1

1. Null, Linda. Lobur, Linda. "The *essentials of Computer Organization and Architecture*". Jones *and* Barllet *Computer Science*. Estados Unidos - Sudbury, Massachusets.

2. http://www.answers.com/topic/massively-parallel - "*Massively Parallel Processing or Massively Parallel Processor",* Copyright © 2007 Answers Corporation.

3. Morrison, Richard S: *"CLUSTER_COMPUTING_THEORY"*, GNU General Public Licence, Abril 2003.

4. Tanenbaum, Andrew S: "*Modern Operating Systems",* Prentice-Hall International, Primera edición, Vrije Universiteit, 1992.

5. http://es.tldp.org/Manuales-LuCAS/doc-manual-openMosix-1.0/doc-manual-openMosix_html-1.0/.

6. http://clusters.fisica.uson.mx/. – *"Clusters de Linux",* Departamento de Física Universidad de Sonora Hermosillo – MEXICO, Ultima actualización: 28 de Septiembre de 2002

7. http://www.consol.org.mx/2002/ponencias/conferencias/Edgar_Govea_-_Clusters.html, Empezando con *Clusters.*

8. Pfister, Gregory F: "*In Search of Clusters"*, Prentice-Hall, 1998.

9. BUYYA, Rajkumar: "*High Performance Cluster Computing*". Prentice Hall, Primera Edición, Estados Unidos, 1999

10. HARGAUGH, Logan: "*Building High-Performance Linux Clusters White Paper*", 2004.

11. http://www.hispatech.com/articulos/html/ibap/infiniband/pag5.php, *Interconexiones y buses de altas prestaciones: Infiniband".*

12. http://dac.escet.urjc.es/docencia/LAAC/practica-monitor-2005-06.pdf, última revisión 22/03/2007.

13. http://gridbus.cs.mu.oz.au/~raj/papers/SSI-CCWhitePaper.pdf*, "SINGLE SYSTEM IMAGE (SSI)",* Universidad Politécnica de Catalunya – Espana.

14. http://es.wikipedia.org/wiki/Middleware, "*Middleware"*, Última modificación 12 de Abril de 2007.

15. Sloan Joseph D: *" High Performance Linux Clusters with OSCAR, Rocks, OpenMosix, and MPI"*, O'Reilly, November 2004, ISBN: 0-596-00570-9.

16. http://www.linux-ha.org/, *"The High Availability Linux Project"*, última actualización 24 de Abril del 2007.

17. http://es.tldp.org/Manuales-LuCAS/doc-manual-openMosix-1.0/doc-manual-openMosix_html-1.0/.

18. http://www.linuxvirtualserver.org/Documents.html, "*Open Mosix version 1.0"*, última actualización 06 de Septiembre del 2004.

19. Sterling, Thomas: "Beowulf Cluster Computing with Windows", MIT Pres Cambridge, Londres.

20. www.emulex.com, "*CLAN".*

21. www.myri.com, "*Myricom",* última actualización 21 de Noviembre del 2006.

CAPÍTULO 2
ADMINISTRACIÓN, SISTEMAS DE ARCHIVOS Y BALANCEO DE CARGA.

2.1 SISTEMAS DE ARCHIVOS. [2][3][4][5][6][7][8][9][10]

2.1.1 INTRODUCCIÓN

Un sistema de archivos es una estructura lógica que permite alojar a directorios y archivos. Un archivo sirve para almacenar datos permanentemente, y a la vez permite manipular los datos (abrir, leer, cerrar, almacenar, entre otras). Los archivos se organizan en estructuras tipo árbol, donde los nodos intermedios son directorios capaces de agrupar a otros archivos. Los archivos y directorios necesitan de un sistema que los organice y gestione; en este sistema es donde se almacenan los archivos, a su vez es un sistema jerárquico de directorios, subdirectorios y archivos.

Linux soporta un gran número de sistemas de archivos, tal es el caso ext2, ext3, ReiserFS e IBM JFS, existen también soporte para sistemas de archivos propios de de cada sistema operativo como el de Windows 9X/ME que usa el vfat (virtual FAT), Windows NT/2000 que utiliza el sistema NTFS, el sistema HPFS de MAC, entre otros.
Todos estos sistemas de archivos poseen sus propios mecanismos de almacenamiento de datos y de metadatos15 en el disco. Bajo un sistema Linux/UNIX la capa de abstracción VFS (Virtual Filesystem Switch) es la que se encarga de la funcionalidad y tareas de almacenamiento de los sistemas de archivos.

---

[15] Los metadatos son estructuras que mantienen la organización de los datos sobre el disco para mantenerlos accesibles todo momento.

2.1.2 TIPOS DE SISTEMAS DE ARCHIVOS.

2.1.2.1 Sistemas de archivos.

2.1.2.1.1 ext2.

Sistema de archivos común de Linux Red Hat, generalmente viene instalado por defecto en las diferentes versiones[16]. Su funcionamiento se basa en la creación de bloques al momento de crear una partición, en esta se instala el sistema de archivos, para lo cual divide la partición en bloques de 1024 bytes cada uno (por defecto[17]). El sistema ext2 tiene una estructura que se resume en la Figura 2-1.

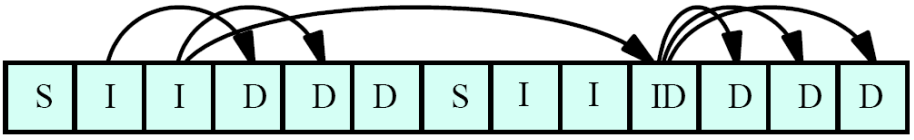


Figura 2-1 . Estructura del sistema de archivos ext2[1].

Cada bloque esta constituido por la siguiente estructura:

- **Superbloque.** Es el primer bloque del sistema de archivos, contiene información de control relacionada al resto de sistemas de archivos, se duplica en cada grupo de bloques para permitir fácilmente recuperar la información del sistema en caso de que los archivos se encuentren corruptos. El superbloque contiene información relacionada al último montaje[18], tamaño del bloque, punteros para i-nodos libre,

---

[16] Las versiones actuales de Linux Red Hat instalan la versión ext3 que no es sino una extensión al sistema de archivos ext2.

[17] Pueden existir bloques de 2048 o 4096 bytes, el tamaño de cada bloque depende del uso que se le vaya a dar.

[18] "*En sistemas Unix un dispositivo físico se representa mediante un archivo especial bajo el directorio /dev. Para poder tener acceso a la información del dispositivo físico es necesario definir un punto de montaje (considerado un directorio dentro del sistema de archivos) mediante el comando* ***mount***". **Fuente:** Proyecto de titulación, "*Clusters de Computadores Personales con Linux*", Mejía, D., Fernández, D. Página 201. Escuela Politécnica Nacional-Quito. Octubre 2005.

punteros para bloques libres, puntero para la raíz del sistema de archivos. En la figura que se muestra a continuación se indica la ubicación del superbloque en un sistema de archivos ext2:

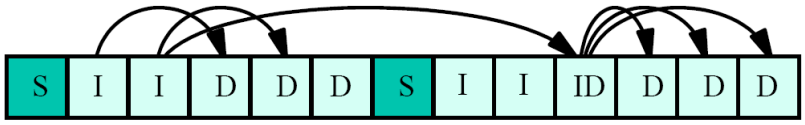


Figura 2-2. Ubicación del Superbloque en ext2.[1]

Como se puede observar en la Figura 2-2, cada bloque posee su respectivo superbloque denotado con una letra **S**, este a su vez posee los *i*-nodos denotados por la letra **I** los cuales apuntan hacia los bloques de datos de notados por la letra **D**, solo si se encuentran en un mismo *i-nodo*; sin embargo pueden apuntar a bloques indirectos los cuales se denotan con **ID**, solo si los datos a leer se encuentran en *i*-nodos diferentes.

- **Bloque Directo (i-nodo).** Bloque que contiene información relacionada a cada archivo individual, dicha información puede corresponder al propietario del archivo, grupo al que pertenece, fecha y hora de creación, modificación y último acceso, también dispone de un grupo de punteros a los bloques de los discos que almacenan información de los archivos. En la Figura 2-3 se muestra la posición de los *i*-nodos dentro de un bloque.

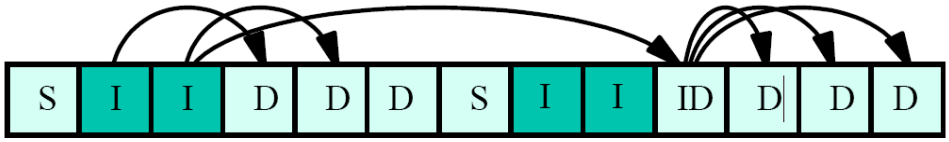


Figura 2-3. Posición de i-nodos en un bloque ext2[1]

- **Bloque Indirecto.** Es apuntado por el *i*-nodo y le proporciona información relacionada al número de bloques de datos y su respectiva ubicación.

- **Bloque de datos.** Utilizados para almacenar los datos contenidos en los archivos y los directorios.

2.1.2.1.2 ext3.

El sistema de archivos ext3 basa su funcionamiento en ext2. El sistema de archivos ext3 utiliza todas las características y funcionalidad de ext2; por otra parte si se desea se puede volver a utilizar ext2 en lugar de ext3 perdiendo las nuevas funcionalidades que presta éste último.

El sistema de archivos ext3 se diferencia del ext2 en que posee un archivo de registro adicional. En este archivo se listan y registran las operaciones que se van a realizar con los metadatos antes de llevarlas a cabo y con ello convertirse en un sistema *journaling* o tolerante a fallos. De este modo si durante el arranque se comprueba que el sistema de archivos está en un estado inconsistente, se puede consultar esta bitácora para ver que operaciones se realizaron cuando el sistema se colapsó, y el análisis y reparación de las estructuras de datos del disco se centran únicamente en esas zonas del disco. Por otra parte los datos que estuvieran a medio escribir en el momento de la detención del sistema se pierden pero se consigue volver al estado consistente inmediatamente anterior en poco tiempo.

El funcionamiento de ext3 es similar al ext2 es decir usa la asignación basada en bloques y la búsqueda secuencial de directorios. ext3 reserva un *i*-nodo especial del ext2 para almacenar el registro, pero los datos de este pueden esta en cualquier otro bloque del sistema de archivos.

En el registro se graban tres tipos de bloques:

- **Meta-información.** Proporciona información del metadato que se está actualizando, además almacena los cambios que se realizaron en el sistema de archivos.

- **Bloques descriptores.** Describe a los otros bloques del registro para que luego sean copiados en el sistema principal, los cambios en estos bloques es escriben antes que los de meta-información.

- **Bloques de Cabecera.** Describe la cabecera y la cola del registro, además proporciona el número de secuencia para garantizar el orden de escritura el momento de recuperar los datos del sistema de archivos.

2.1.2.1.3 Reiserfs

Sistema de archivos diseñado e implementado por Hans Reiser y su equipo de desarrolladores de la empresa *Namesys*, las versiones ReiserFS 3.6.x viene incluida en la versión de *kernel* 2.4.x y 2.6.x. ReiserFS busca convertirse en un buen sistema de archivos que proporcione un entorno común para las acciones a realizarse sobre éste y con ello lograr un mejor rendimiento del sistema.

Basa su funcionamiento en la utilización de árboles balanceados[19] para la organización de los objetos del sistema de archivos. Estos objetos contienen información relacionada a los permisos, modificación, tiempos de creación, acceso y datos del archivo, haciendo analogía con los sistemas ext2 y ext3, ReiserFS contiene la misma información que los *i*-nodos, directorios y bloques de datos. ReiserFS a los objetos los llama de tres maneras:

- **Ítems de datos (*Stat data*).** Contiene información de la meta-información, es decir proporciona información de los meta-datos.

- **Ítems de directorio.** Proporciona información relacionada a todos los directorios del sistema de archivos.

- **Items directos.** Apuntan a la ubicación de los datos de los archivos que se encuentran en un mismo árbol, es decir son los propios datos del archivo.

- **Ítem indirectos.** Apuntan a la ubicación de los datos que se encuentran almacenados dentro de otro árbol.

---

[19] Ver sitio Web: http://cslibrary.standford.edu/110/. En éste sitio se encuentra la información completa de técnicas que involucran árboles (balanceados, B+ y otros).

ReiserFS provee tolerancia a fallos mediante la utilización de la técnica de *journaling* de sólo la meta-información de los archivos más no de los bloques de datos en si.

Este sistema de archivos se caracteriza por tratar de almacenar la meta-información de los archivos (*stat ítems y direct/indirect ítems)* junto a los datos para con ello lograr tener un pronto acceso al sistema de archivos. ReiserFS realiza un empaquetado de colas[20] donde se almacena la meta-información.

La ventaja principal de este sistema de archivos respecto a los sistemas ext2 y ext3 radica en la forma de leer los archivos, ReiserFS mejora el rendimiento de la lectura de archivos mediante el uso de la técnica de árboles balanceados de lista de directorios lineales, mientras que ext2 y ext3 lo realizan mediante el uso de *i*-nodos con lectura secuencial.

Una de las desventajas de esta técnica radica en el consumo recursos de CPU al momento del empaquetamiento de las colas, afectando con ello al rendimiento del sistema, de este modo el sistema se vuelve lento. Debido a que ReiserFS se encuentra aun en desarrollo se espera que este problema sea solucionado en las siguientes implementaciones.

2.1.2.1.4 JFS

JFS (*Journaled File System)* sistema de archivos desarrollado por IBM, el cual posee un buen comportamiento en sistemas que poseen altas velocidades de transferencia. Basa su funcionamiento en *extents* o secuencias de bloques contiguos asignados a un archivo como una sola unidad, los bloques pueden ser de diferente tamaño ya sea de 512, 1024, 2048 y 4096 bytes, siendo éste último el tamaño por defecto. El direccionamiento se basa en árboles B+ de descriptores de *extents* lo cual le da un buen rendimiento al sistema de archivos.

JFS al igual que los otros sistemas de archivos, posee un registro donde constan todas las acciones realizadas en la meta-información de los archivos. Estos archivos

[20] Las colas son archivos más pequeños que un bloque normal y se ubican en espacios libres que los archivos dejan al no llenar los bloques completos.

son de tipo síncrono, es decir el sistema de archivos se bloquea mientras se realiza una transacción de almacenaje en el disco y no permite realizar otra operación mientras el archivo no haya sido guardado, esto garantiza que si se tiene un reinicio inesperado, la información podrá ser recuperada.

2.1.2.1.5 XFS

Sistema de archivos desarrollado por SGI como un sistema *journaling* para Linux, es un sistema muy utilizado para sistemas de archivo de alta velocidad de transferencia al igual que JFS es muy utilizado para aplicaciones en tiempo real como video y generación de imágenes los cuales manejan sistemas de archivos muy grandes.
XFS basa su funcionamiento en árboles B+ con *extents* que utiliza registros de tipo asíncronos en donde el sistema de archivos principal no se bloquea y continua haciendo otras operaciones de escritura y lectura en el sistema de archivos, mientras el archivo se almacena, los registros asíncronos utilizan *caching* o caché para almacenar las acciones que se estén haciendo en el sistema de archivos y, así mantener activo el sistema siempre; el reinicio inesperado del sistema haría que la información no almacenada se pierda; ésta es una de las desventajas de los registros asíncronos.

El uso de *extents* permite realizar un reagrupamiento de espacio luego de eliminar un archivo, permite agrupar en cada bloque más espacio de disco (desde 512 bytes a 64 KB) para el almacenamiento de la información; se optimiza el tiempo de acceso al CPU al no tener que buscar los archivos en bloques contiguos. XFS permite implementar listas de acceso o ACL (*Access Control List*) en el sistema de archivos, dando con ello un mayor grado de seguridad en términos de autorización o permisos.

2.1.2.1.6 PVFS

Sistema de archivos paralelo virtual (PVFS – *Parallel Virtual File System*), es una implementación de código abierto que fue desarrollado por el Laboratorio Nacional Argonne y la Universidad de Clemson de los Estados unidos, fue desarrollado para sistemas operativos Linux, muy utilizado en *clusters* de computadores. PVFS es un sistema que permite a diversas aplicaciones almacenar datos en servidores distribuidos en la red, es decir permite dividir un archivo en varias partes las cuales

serán distribuídas entre los servidores de I/O a los cuales los clientes accederán; con ello se logra tener un buen rendimiento del sistema cuando se trata de escribir archivos de gran tamaño.

PVFS para la comunicación entre los servidores utiliza el protocolo TCP/IP, con lo que se evita el utilizar sistemas de paso de mensajes. Su funcionamiento se basa de acuerdo a tres componentes:

- **Servidor de Metadatos.** Servidor que contiene información relacionada a permisos, pertenencia y localización de los datos. A este servidor en primera instancia acceden los clientes para realizar cualquier cambio en los datos ya sea para copiar, mover, borrar, abrir, cerrar archivos que están almacenados en los servidores de I/O.

- **Servidores de I/O.** Servidores donde se almacenan todos los datos del sistema de archivos, el tamaño total o espacio de almacenamiento es la suma de todo el espacio que ofrecen en conjunto los servidores de I/O.

- **Nodos Clientes.** Son los usuarios del sistema PVFS.

Se debe tener presente que PVFS permite que los nodos clientes se conviertan en servidores de I/O y viceversa, pero esto no se cumple para el servidor de metadatos el cual no puede convertirse en nodo cliente y únicamente un servidor de I/O puede reemplazarlo.

2.1.3 REPLICACIÓN DE ARCHIVOS

2.1.3.1 Introducción.

La replicación de archivos es una herramienta fundamental para lograr una alta disponibilidad de los servidores de un cluster, esta característica permite a mencionados servidores trabajar de forma paralela dando la imagen de un solo sistema e incluso permite sustituir unos a otros en sus funciones. Con este mecanismo se busca tener réplicas exactas unos de otros, los cuales sirvan todos el

mismo contenido, para ello se debe encontrar la manera de lograr réplicas de manera automática de forma transparente para el usuario.

Para lograr la replicación de archivos se lo puede hacer de dos maneras: la primera se realiza mediante la replicación física de archivos, donde cada servidor tendrá una misma copia de todos los datos en su disco duro, el segundo método se realiza mediante sistemas de archivos distribuidos, donde existe un servidor de archivos al cual acceden el resto de servidores acceden a través de la red.

A continuación se analizan cada uno de los métodos para realizar la replicación de archivos.

2.1.3.2 Replicación física de archivos.

Mecanismo de replicación de archivos de manera manual o automática con lo cual se distribuye los datos a todos los servidores que componen el *cluster*, uno de lo métodos más populares es *rsync*, el cual se analizará a continuación:

2.1.3.2.1 rsync

Algoritmo[21] utilizado para realizar copias entre dos servidores, este proceso se caracteriza por sincronizar archivos y directorios sobre los cuales se transfieren únicamente las modificaciones que a éstos se hayan realizado, con ello se logra tener al día copias idénticas de los directorios del sistema de archivos presentes en los servidores que lo conforman. Con el algoritmo *rsync* se puede realizar varias actividades tales como:

- Copiar en su totalidad ficheros, directorios, sistemas de archivos manteniendo una sincronización adecuada mediante la cual se puede realizar copias únicamente de los cambios realizados.

[21] Ver sitio Web: http://*rsync.samba.org.*

- Realizar la copia de datos de manera segura, ya que puede utilizar *ssh*[22], algoritmo para cifrar el tráfico de datos.

2.1.3.3 Replicación mediante un sistemas de archivos distribuidos.

Un sistema de archivos distribuido es aquel que almacena archivos en uno o varios servidores, a los cuales los clientes acceden de manera remota. Los clientes ven al sistema de archivos como si fuese local y el acceso se logra mediante alguna infraestructura de red.

Existen varias ventajas de tener un sistema de archivos distribuido: los archivos están más accesibles, compartir los archivos desde una localización única es más sencillo que distribuir copias de los archivos a todos los clientes, las copias de respaldo y la seguridad son más fáciles de manejar cuando sólo hay que tener en cuenta a los servidores, los servidores pueden ofrecer un gran espacio de almacenamiento.

También existen algunos inconvenientes al mantener un sistema de archivos distribuido: el hecho de transferir un gran número de archivos por la red puede generar problemas como latencia, cuellos de botella, entre otros; en definitiva puede generar decremento en la eficiencia de la red. De igual manera los aspectos de seguridad pueden ser muy críticos si no se tiene políticas que mantengan a la información íntegra, es decir se debe tomar en cuenta mecanismos de autenticación para usuarios y seguridades a nivel de red.

A continuación se explican algunos sistemas de archivos muy conocidos y útiles.

2.1.3.3.1 NFS

EL sistema de archivos de red (NFS – *Network File System*) fue creado por SUN y publicados en el año de 1985, es un sistema que ofrece soluciones de seguridad (autenticación basada en KERBEROS[23]), alto rendimiento y acceso transparente al

---

[22] SSH (*Secure Shell)* es una técnicas de cifrado que hacen que la información que viaja por el medio de comunicación vaya de manera no legible y ninguna tercera persona pueda descubrir el usuario y contraseña de la conexión ni lo que se escribe durante toda la sesión.

[23] Sitio Web: http://cryptnet.net/fdp/admin/kerby-infra/en/kerby-infra.html.

sistema de archivos. NFS es un estándar abierto basado en la arquitectura Cliente/Servidor, la cual es útil para compartir archivos por la red independientemente del sistema operativo del cliente o del servidor y, a la vez puede instalarse en cualquier plataforma.

El uso de NFS ofrece varias ventajas como:

- La existencia de archivos a los que se pueden acceder de manera simultánea por varios clientes a través de la red.
- Reduce costos de almacenamiento debido a que tiene los archivos compartidos en la red y no necesita de espacio en disco local.
- Realiza el montaje de los sistemas de archivos a los usuarios de manera transparente.
- El acceso a los archivos remotos es transparente para el usuario.
- Soporta ambientes heterogéneos, es decir que se puede instalar en aquellas infraestructuras que no disponen de arquitecturas de iguales o similares características.

NFS consta de un programa cliente y servidor. El programa servidor comparte los archivos mediante el proceso de exportación, mientras que el cliente accede al sistema de archivos compartidos añadiéndolos a su sistema local mediante el proceso de montaje. Los protocolos[24] de NFS proporcionan el medio de comunicación entre los procesos de exportación y montaje sobre la red.

En el servidor para realizar el proceso de exportación se debe editar un archivo de configuración (**`/etc/exports`**) el cual consta de tres partes:

1. Ruta completa del archivo a exportar.
2. Máquinas que son autorizadas para acceder al archivo exportado.
3. Restricciones de acceso.

---

[24] Los protocolos de NFS utilizan llamadas a procedimientos remotos RPC (*Remote Procedure Call*) que a su vez se basan en IPC (*Inter Process Comunication*), que no es sino la capacidad del sistema operativo para permitir la comunicación entre procesos, sea que estos se ejecuten de manera local o remota.

```
/mnt/cdrom *(ro)
/usr/games *(rw)
/home *.local.domain(rw)
/budgets *.accounting(ro) trusty(rw,no_root_squash)
```

Figura 2-4. Ejemplo de archivo /etc/exports.

Cuando el servidor arranca el sistema, este archivo de configuración es leído y se informa al *kernel* de los tres aspectos que conlleva la exportación de un archivo.

El cliente es encargado de montar un sistema de archivos remoto, no hace una copia de este, sino realiza un montaje por medio de una serie de llamadas de procedimiento remoto con el que se habilita al cliente el acceso al sistema de archivos. El cliente NFS tiene la característica de montar y desmontar el sistema de archivos de acuerdo a las necesidades, pero todo el proceso se realiza de manera transparente para el usuario.

2.1.3.3.2 AFS

El sistema AFS (*Andrew File System)* es un sistema desarrollado por la Universidad de Carnegie Mellon. OpenAFS es la implementación de código abierto del sistema de archivos Andrew el cual basa su funcionamiento en la arquitectura Cliente/Servidor que permite migrar los datos de forma transparente.

AFS es un sistema de archivos distribuido, el cual permite a Clientes/Servidores compartir recursos a nivel de red LAN o WAN, AFS es un sistema diseñado para trabajar sobre el protocolo TCP/IP.

Este sistema de archivos permite a los clientes acceder a sus archivos desde cualquier lugar, debido a que AFS está optimizado para redes de área amplia (WANs). AFS además proporciona mecanismos de seguridad mediante el uso de Kerberos para la autentificación de usuarios y listas de control de accesos (ACLs) para restringir el acceso de los usuarios a los directorios.

AFS para su funcionamiento requiere de dos componentes (procesos):

- Venus:
  - Se ejecuta en los clientes.
  - El sistema operativo redirecciona las peticiones sobre archivos compartidos.
  - Realiza las traducciones de nombres de ficheros.

- Vice:
  - Se ejecuta en los servidores.
  - Procesa solicitudes remotas de los clientes.

2.1.3.3.3 PVFS

Sistema de archivos paralelo virtual (PVFS – *Parallel Virtual File System*), es una implementación de código abierto que fue desarrollado por el Laboratorio Nacional Argonne y la Universidad de Clemson de los Estados unidos, fue desarrollado para sistemas operativos Linux, muy utilizado en *clusters* de computadores. PVFS es un sistema que permite a diversas aplicaciones almacenar datos en servidores distribuidos en la red, es decir permite dividir un archivo en varias partes las cuales serán distribuídas entre los servidores de I/O a los cuales los clientes accederán; con ello se logra tener un buen rendimiento del sistema cuando se trata de escribir archivos de gran tamaño.

PVFS para la comunicación entre los servidores utiliza el protocolo TCP/IP, con lo que se evita el utilizar sistemas de paso de mensajes. Su funcionamiento se basa de acuerdo a tres componentes:

- **Servidor de Metadatos.** Servidor que contiene información relacionada a permisos, pertenencia y localización de los datos. A este servidor en primera instancia acceden los clientes para realizar cualquier cambio en los datos ya sea para copiar, mover, borrar, abrir, cerrar archivos que están almacenados en los servidores de I/O.

- **Servidores de I/O.** Servidores donde se almacenan todos los datos del sistema de archivos, el tamaño total o espacio de almacenamiento es la suma de todo el espacio que ofrecen en conjunto los servidores de I/O.

- **Nodos Clientes.** Son los usuarios del sistema PVFS.

Se debe tener presente que PVFS permite que los nodos clientes se conviertan en servidores de I/O y viceversa, pero esto no se cumple para el servidor de metadatos el cual no puede convertirse en nodo cliente y únicamente un servidor de I/O puede reemplazarlo.

2.1.3.3.4 CODA

Sistema de archivos distribuido, desarrollado por la Universidad de Carnaige Mellon desde 1987, cuyo desarrollo fue basado en el sistema de archivos AFS2 (*Andrew File System versión 2)*.

CODA es un sistema de archivos que brinda muchas utilidades, las cuales lo hacen muy atractivo para ser utilizado en un *cluster*, en especial para brindar servicios de alta disponibilidad.

Dentro de CODA se debe considerar varias terminologías que son usadas dentro de este sistema:

- Espacio Único de Nombres.- Todo el sistema de archivos CODA debe estar bajo un único directorio (*/coda*), todas las carpetas compartidas por parte de los servidores de archivo se encuentran en un mismo directorio, logrando con ello tener un aspecto unificado de todos los directorios compartidos.

- Celda CODA.- Formada por un conjunto de servidores caracterizados por tener la misma configuración. Una celda puede estar conformada con un servidor principal o servidor maestro denominado *SCM* (*System Control Machine*) que es el encargado de controlar el sistema y es quien posee la copia principal del sistema de archivos, además *SCM* es el único que tiene la potestad de modificar el sistema de archivos y a la vez distribuirlo al resto de servidores (clientes Coda). Un cliente Coda sólo puede pertenecer a una sola celda.

- Volúmenes CODA.- Los servidores de archivos se caracterizan por agrupar los archivos compartidos en volúmenes, un volumen puede ser más grande que un directorio pero más pequeño que una partición, se caracteriza por tener una raíz principal con varios subdirectorios. Los volúmenes se montan bajo el directorio /coda y forman un subárbol de este, pudiendo existir la posibilidad de montar un volumen dentro de otro. El grupo de servidores que sirven un mismo volumen se denomina *VSG* (*Volume Storage Group*).

- Puntos de montaje de volúmenes.- El volumen raíz se monta directamente en el directorio principal */coda*, el resto de volúmenes pueden montarse dentro del árbol */coda* mediante el uso del comando *cfs mkmount*.

- Almacenamiento de datos.- El almacenamiento de los archivos compartidos no se hace directamente en el disco local, sino se realiza un almacenamiento de archivos compartidos bajo el directorio */vicepa*, en el que la información se guarda en forma de archivos que se distribuyen a lo largo de un árbol de directorios codificado numéricamente.

- Memoria Virtual Recuperable (*RVM - Recoverable Virtual Memory)*.- Coda usa RVM para administrar sus metadatos (propietarios de archivos, listas de control de acceso, fecha de creación; entre otras). Estos datos son almacenados en un archivo RVM almacenado en disco y que es mapeado a la memoria únicamente para arrancar el momento de un reinicio inesperado del sistema. Las modificaciones se hacen en la RVM y son escritas en el archivo de log RVM cuando una transacción es realizada. El archivo de log contiene los datos que se han transmitido/modificado y que todavía no han sido incorporados al archivo de datos del disco.

- Datos del cliente.- Coda a nivel del cliente posee un caché local sobre el cual se encuentra almacenada toda la información relacionada a los archivos que los clientes accedieron recientemente, logrando con ello tener un acceso rápido a los datos y a la vez poder acceder a estos aunque exista una desconexión entre el cliente y el servidor.

- Validación.- Mecanismo utilizado para validar los datos del caché local respecto a los datos existentes en los servidores, logrando tener una versión de los archivos actualizada.

- Autentificación-. La autentificación que utiliza CODA es similar al protocolo de autentificación *Kerberos*[25], donde el cliente genera una palabra de paso la cual es dada al servidor y este a su vez le proporciona una clave de entrada (testigo), con ello ambos tienen la misma información para realizar un correcto proceso de autentificación.

- Protección.- La protección del sistema de archivos compartido, a más de realizarla mediante un adecuado sistema de autentificación, utiliza un mecanismo de listas de control de acceso (ACL, Access Control Lists) con las que, el servidor puede discriminar que cliente tiene acceso a un determinado recurso del sistema de archivos compartido una vez que ha sido autentificado.

El protocolo CODA a nivel de servidores se sustenta sobre tres servicios:

- Servidor de archivos. El principal programa del servidor de archivos Coda es *codasrv,* el cual juntamente con el proceso *Venus*[26] son los que realizan todo el intercambio de datos (archivos) entre las máquinas.

- Servidor de Autentificación. El servidor de autentificación *auth2* se ejecuta en todos los servidores, la funcionalidad principal es de validar la

---

[25] *Kerberos. P*roceso de autenticación desarrollado en MIT (*Massachusetts Institute of Technology*) y diseñado por Miller y Neuman en el año de 1987. Protocolo de distribución de claves, donde cada usuario y servidor tendrán su propia clave, las cuales deben estar contenidas en las bases de datos del servidor de autentificación *Kerbero*s. Un usuario posee una clave que será derivada de su contraseña y estará encriptada, mientras que en el caso del servidor, la clave se generará aleatoriamente. Los servicios de red que requieren autenticación y los usuarios que requieran estos servicios, se deben registrar con Kerberos.

[26] Proceso *Venus.* Proceso que almacena aquella información que necesita el cliente, como archivos y sus atributos. Si el fichero ha sido modificado y cerrado, entonces Venus actualiza los servidores enviándoles el nuevo fichero. Cualquier otra operación que modifique el sistema de archivos se propagará también a los servidores.

autenticidad de los usuarios que acceden al sistema de archivos Coda y controlar sus tokens de identidad. Las contraseñas utilizadas para el proceso de autentificación sólo pueden ser modificadas en el SCM.

- Servidor de Actualización. Conformado por el proceso *updateclnt* el cual trabaja juntamente con el proceso *updatesrv* quien se ejecuta en el SCM para mantener actualizadas las copias de las bases de contraseñas y de información en todos los servidores con la copia original en el SCM. Para lograr este propósito el demonio *updateclnt* comprueba cada cierto tiempo si los ficheros del SCM han sido actualizados. Las actualizaciones dependen exclusivamente del periodo de comprobación de *updateclnt*.

Es importante aclarar que un mismo servidor puede ofrecer los tres servicios a la vez, o tan sólo alguno de ellos. En cualquier caso, el demonio *updateclnt se* debe ejecutarlo siempre para mantener actualizadas las copias de los archivos del sistema y de la base de datos de contraseñas.

El sistema de archivos Coda ofrece un mecanismo de seguridad para sus recursos, es decir con él provee un adecuado mecanismo de identificación y autorización de acceso por parte de los usuarios.

El mecanismo mediante el cual los servidores Coda presentan los datos es mediante estructuras denominadas volúmenes. Un volumen posee una estructura de directorios la cual es utilizada para representar un conjunto de archivos en un servidor Coda. La información contenida en estos volúmenes se guarda dentro del servidor Coda como metadatos en un área de almacenamiento especial.

Los volúmenes se pueden montar dentro de sistemas de archivos Coda ya montados, en una celda Coda siempre existe un volumen raíz /Coda sobre el que se montan otros volúmenes que aparecen como subdirectorios nuevos dentro del sistema de archivos Coda.

Una de las características principales de un sistema de archivos Coda es la replicación de volúmenes, con esto un mismo volumen puede almacenarse en varios

servidores, logrando con ello solucionar problemas de acceso de los clientes cuando un servidor falla, y cuando este se restituye el sistema se encarga de propagar automáticamente los cambios realizados en el volumen.

Además es importante mencionar que el sistema permite a los clientes trabajar con el caché local de los clientes Coda, con lo cual se puede trabajar en estado desconectado, donde las operaciones solicitadas por las aplicaciones sobre los archivos en la cache son guardadas por el cliente, de forma que cuando éste se reconecte al sistema propaga los cambios realizados localmente a los servidores que contengan los archivos.

2.1.3.3.5 Otros

- **NAS (*Network Attached Storage*)**. Mecanismo caracterizado por tener el dispositivo de almacenamiento conectado a la red, encargándose éste de dar acceso a los clientes y servidores a archivos, es decir NAS se dedica a transportar los datos de la red a un disco duro y viceversa. El proceso de almacenamiento de información es independiente al sistema operativo que se esté utilizando.

- **SAN.** La red de área de almacenamiento (SAN – *Storage Area Network*), red caracterizada por interconectar servidores, discos de almacenamiento, dispositivos NAS; entre otros. La interconexión se realiza mediante la utilización de técnicas de red de alta velocidad por lo general basadas en canales de fibra óptica con lo cual se tiene un alto rendimiento en el acceso a datos.

- **InterMezzo.** Sistema de archivos distribuido para Linux, diseñado por la Universidad Carnegie Mellon e inspirado en el sistema de archivos Coda. Se incluye el soporte para InterMezzo a partir de la versión 2.4.5 del *kernel* de Linux.

  InterMezzo posee su propio sistema de archivos, los cuales se almacenan de manera local en una partición del disco duro de cada ordenador. Para realizar

el almacenamiento de archivos de manera local requiere de alguno de los sistemas de archivos disponibles como es el caso de ext2, ext3, ReiserFS, entre otros. A partir de este momento, todas las operaciones que se realice se almacenaran en éste sistema de archivos y luego se actualizarán en el sistema de archivos original.

2.1.4 ESTRUCTURA DE UN SISTEMA DE ARCHIVOS EN LINUX.

El sistema de archivos de Linux es una estructura empleada por el sistema operativo para almacenar información en los dispositivos físicos como un disco duro, CD-ROM, DVD; entre otros. En los archivos se puede almacenar cualquier tipo de información, ya se imágenes, música, texto; entre otras. Linux maneja únicamente tres tipos de archivos en su estructura, los archivos propiamente dichos, directorios o carpetas quienes agrupan a los archivos de manera estructurada y los archivos especiales quienes representan a los dispositivos conectados al servidor como es el caso de una impresora.

La estructura del sistema de archivos Linux es jerárquica, tiene una organización en forma de un árbol, cuya raíz principal es / sobre la cual se desprenden todos los archivos y directorios que componen el sistema de archivos.
Los sistemas de archivos Linux siguen todas las convenciones de Unix, logrando con esto tener una compatibilidad con el resto de sistemas operativos basados en éste. El directorio principal en el sistema de archivos Linux es la raíz o *root* representada por /, bajo este directorio se encuentran el resto de directorios y archivos a los cuales puede acceder el sistema operativo y que se detallan a continuación:

- **/.** Raíz del sistema de archivos.
- **/dev.** Contiene archivos del sistema que representan a los dispositivos físicos conectados al servidor.
- **/home.** Contiene los archivos personales de los usuarios.
- **/etc.** Directorio reservado para los archivos de configuración del sistema. Bajo este deben aparecer otros dos subdirectorios:

- /**mnt.** Este directorio se ha provisto para que el administrador pueda montar temporalmente sistemas de archivos cuando lo necesite. El contenido de este directorio es un asunto local y no debe afectar la manera en la cual se ejecuta ningún programa.
- **/lib.** Contiene las librerías necesarias para que se ejecuten los programas.
- **/proc.** Contiene archivos especiales que reciben o envían información al kernel del sistema, el contenido de estos no se debe modificar.
- **/sbin.** Contiene programas que son únicamente accesibles al superusuario o *root*.
- **/usr.** Éste es uno de los directorios más importantes del sistema puesto que contiene los programas de uso común para todos los usuarios. Su estructura suele ser similar a la siguiente:

  - **/usr/bin.** Programas de uso general, lo que incluye el compilador de C/C++.
  - **/usr/doc.** Documentación general del sistema.
  - **/usr/etc.** Archivos de configuración generales.
  - **/usr/incluye.** Archivos de cabecera de C/C++ (***.h**).
  - **/usr/lib.** Librerías generales de los programas.
  - **/usr/man.** Manuales accesibles con el comando man.
  - **/usr/sbin.** Programas de administración del sistema.
  - **/usr/src.** Código fuente de programas.

- **/var.** Datos varios como archivos de log (registro de actividades) de programas, bases de datos, contenidos del servidor Web, copias de seguridad; entre otros. Este directorio contiene información temporal de los programas.

2.1.5 TIPOS DE SISTEMAS DE ALMACENAMIENTO.[11][12]

Los sistemas de almacenamiento son de gran importancia a la hora de configurar un *cluster* de alta disponibilidad; permiten asegurar la integridad y confiabilidad de los datos almacenados en los discos de los servidores, esta manera se logra evitar puntos de falla únicos (*SPOF-Single Point Of Failure*) en elementos de almacenamiento y caídas del sistema.

2.1.5.1 Administración de Discos

Los discos de almacenamiento pueden ser configurados de diferente manera ya sea utilizando un arreglo RAID[27] (*Redundant Array of Inexpensive Disks*) o utilizando la configuración LVM (*Logical Volume Management*).

2.1.5.1.1 RAID.

Mecanismo utilizado para dar una mayor rapidez en el acceso a los datos, mayor capacidad y tolerancia a fallos. Como su nombre lo indica RAID consiste en crear un arreglo (*array*) de discos simples de bajo costo (*inexpensive*) para tratarlos como uno solo cuando se requiere el acceso a los datos. Se puede tener sistemas basados en tecnología RAID a nivel de hardware y software.

Los RAID a nivel de hardware se caracteriza por ser un subsistema independiente del *host*, presentándole a este un solo disco o una sola imagen. Para su gestión requiere de un controlador externo cuyo funcionamiento es autónomo.

Los RAID a nivel de software, caracterizados por implementar la gestión de los discos para los diferentes niveles de RAID en el código del *kernel*. En la actualidad las prestaciones de un sistema RAID de software son similares a las de hardware, gracias a la potencia de los CPUs.

2.1.5.1.2 Niveles de RAID[28].

Existen varios niveles de configuración:

- **Lineal.** Configuración que basa su funcionamiento en la utilización de dos o más discos instalados de manera sucesiva. Cuando se llena el primer disco, se utiliza el segundo y así sucesivamente, en esta configuración no se consigue aumento de

[27] Ver sitio *http://es.wikipedia.org/wiki/RAID*.

[28] Ver sitio Web: *http://www.monografias.com/trabajos6/sira/sira.shtml*.

velocidad, seguridad por redundancia, ni respaldo de discos, si se daña uno de ellos se pierde la información almacenada en él. Modo lineal crea un dispositivo virtual de mayor tamaño.

- **RAID 0.** También conocido como *Stripe*. Configuración que consiste en un disco o conjunto de discos en los que la información a ser almacenada es dividida en bloques y cada bloque es grabado en una unidad de disco diferente, en este método no existe redundancia (sin tolerancia a fallos) pero es un mecanismo más rápido al acceder a los dispositivos ya que se están haciendo en paralelo siendo ésta una de las principales razones para utilizar un sistema RAID 0. Los discos deben ser de aproximadamente el mismo tamaño y misma velocidad para obtener rendimientos óptimos. RAID 0 es aplicado en sistemas que requieren gran ancho de banda como edición y producción de imágenes y vídeo. Una representación de lo anteriormente descrito se puede visualizar en la Figura 2-5.

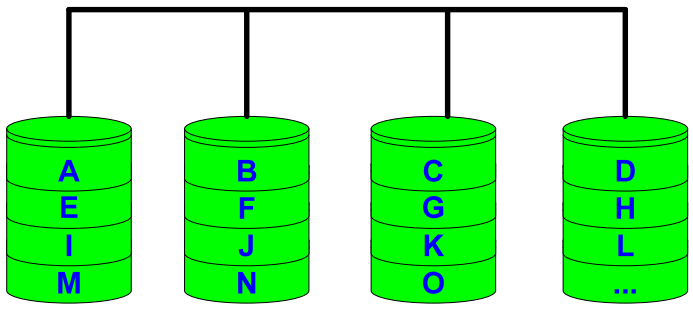


Figura 2-5. RAID nivel 0.

- **RAID 1.** También denominado de espejo o mirroring. Primera configuración que añade redundancia. En esta configuración no se pierde la información, ya que si un disco falla, automáticamente utiliza uno de sus discos disponibles para sustituirlo. Los discos que se están utilizando deben tener el mismo tamaño y su escritura es lenta debido a que se debe replicar la información en todos los discos.

  RAID 1 es aplicado en ambientes requeridos de alta disponibilidad como áreas contables, finanzas, servidores y otros.

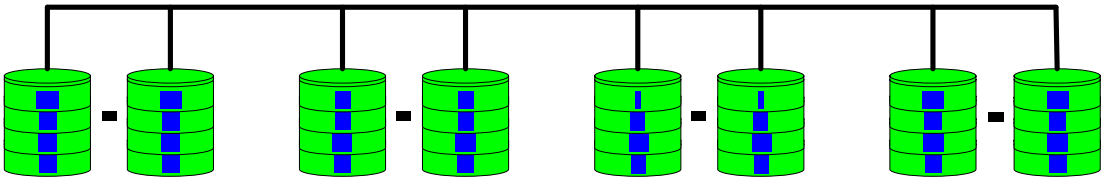


Figura 2-6. RAID nivel 1.

- **RAID 2.** Esquema RAID redundante de nueve discos muy poco utilizado, caracterizado por usar códigos de corrección de errores, los cuales se basan en la utilización de bits de paridad que permiten detectar errores a nivel de bit.

  Una de las principales ventajas por las que fue creada esta configuración es la velocidad de transferencia en la lectura de datos debido a que los discos están trabajando en paralelo y una de sus desventajas está en la escritura ya que debe copiar los datos en todos los discos.

- **RAID 3.** Conocido también como *striping* con paridad o acceso síncrono, consta de varios discos en paralelo y un disco de paridad para corrección de errores y almacenamiento de información de control codificada. La recuperación de datos se consigue calculando el O exclusivo (XOR) de la información registrada en los otros discos, este mecanismo permite proporcionar redundancia de datos. Los datos a almacenarse son divididos en fragmentos los cuales son enviados a los discos que funcionan en paralelo, esta característica permite tener una mayor velocidad de transferencia de datos.

  El incremento de velocidad es muy importante en sistemas que requieren grandes transferencias de datos secuenciales como es el caso de aplicaciones de audio y video. En casos de alta velocidad todos los discos de nivel 3 de RAID participan en cada transacción, atendiendo a cada petición de Entrada /Salida. Por consiguiente el nivel 3 de RAID no es una opción adecuada para operaciones transaccionales, en la que la mayor parte del tiempo se emplea en buscar pequeños registros esparcidos aleatoriamente en los discos. La búsqueda se puede observar en la Figura 2-7.

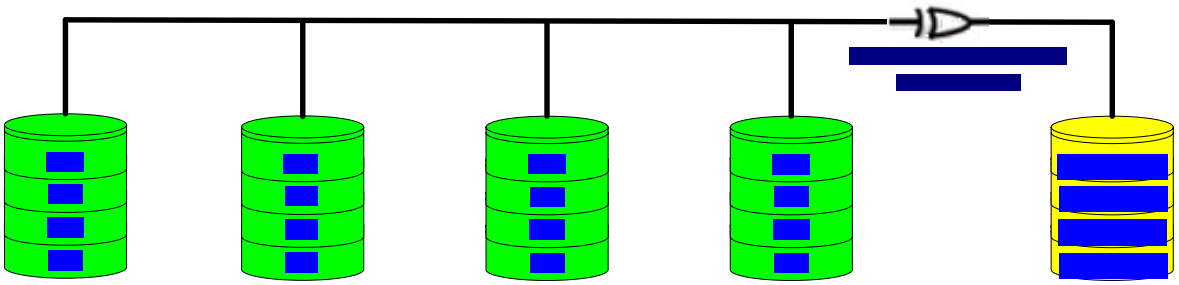


Figura 2-7. RAID nivel 3.

- **RAID 4.** Denominado también acceso independiente con un disco dedicado a paridad, esta configuración necesita tres o más discos para su funcionamiento, los discos son divididos como en RAID 0 y la paridad de información para la división es calculada y almacenada en discos de paridad.

  La capacidad total del conjunto es de (N.1)* R, donde N es el número total de discos existentes en el sistema y, R es el tamaño del disco de menor capacidad, esto en caso de que los discos sean de diferente tamaño.

  La ventaja de utilizar este sistema esta en la velocidad de acceso a los discos debido a su trabajo en paralelo, también está la redundancia de datos que provee el disco de paridad.

  La desventaja de utilizar esta configuración se debe a que la información copiada se mantiene en un solo dispositivo y esta a la vez debe ser actualizada cada vez que se escribe en los otros discos, lo cual se convierte en un cuello de botella haciendo que el sistema se vuelva lento. Otra desventaja se debe a que si se pierde los datos de dos discos o más, es difícil que el disco de paridad los pueda recuperar, lo cual hace que se pierda toda la información.

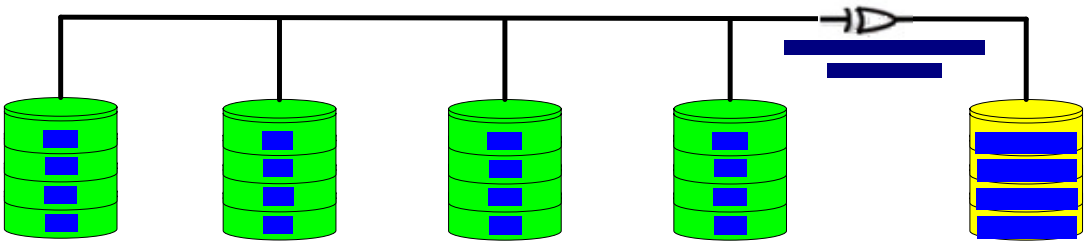


Figura 2-8. RAID nivel 4.

- **RAID 5.** Discos de datos independientes con bloques de paridad distribuidos. Esquema muy utilizado cuando se quiere tener un gran número de discos físicos y alguna redundancia, compuesto de un mínimo de tres discos y discos de repuesto. El tamaño total del conjunto se calcula de idéntica manera que en RAID 4 ((N.1)* R).

  Los datos son divididos en bloques de datos, estos son almacenados en los discos dentro del conjunto, también se generan bloques de paridad que se distribuyen entre los discos existentes en el sistema, logrando con esto tener redundancia que permitirá recuperar la información en caso de un fallo en los datos. Este bloque de paridad debe recalcularse cada vez que exista una nueva escritura en cada disco.

  En este tipo de configuración no se tiene el problema de cuello de botella que existía en los RAID 4, ya que los bloques de paridad no se almacenan en un solo dispositivo, sino que se distribuyen por todos los discos existentes en el sistema, existe mejora en el rendimiento al momento de hacer una lectura ya que estos bloques no son analizados, de modo que se evita el tener una sobrecarga innecesaria que afecte el funcionamiento del sistema.

  Los bloques de paridad son leídos únicamente cuando existen errores en el cambio de la información inicial (CRC o *checksum*) de los datos almacenados, y la finalidad de la lectura es para reconstruirlos. La reconstrucción se realiza en base al uso de cada bloque de datos y del bloque de paridad correspondiente a mencionados bloques, esto para reconstruir el sector erróneo. Si un disco queda fuera, el grupo de discos restantes combinan

matemáticamente sus bloques de datos y de paridad para reconstruir la información perdida, la computadora no se entera que un disco ha fallado, el proceso de escritura y lectura es normal, salvo una leve modificación en el rendimiento del sistema.

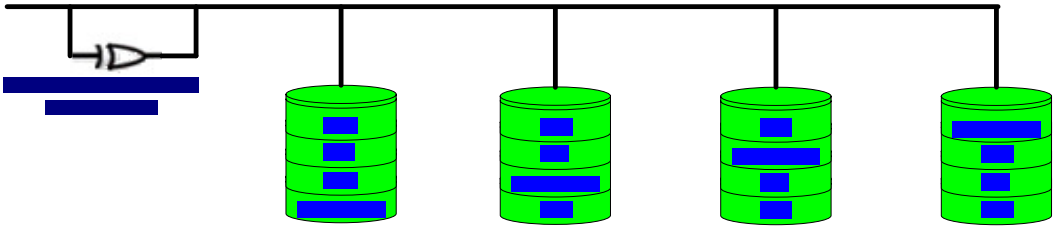


Figura 2-9. RAID nivel 5.

- **RAID 6.** Similar a RAID 5, varía de RAID 5 debido a que bloques enteros de datos son grabados en los discos y se genera un segundo esquema de paridad distribuido en los diferentes discos logrando con ello una alta tolerancia a fallos y caídas de disco.

  RAID 6 requiere de un mínimo de tres discos para su funcionamiento, mientras el número aumente el rendimiento será mejor y con él mayor redundancia en paridad.

  Los discos se organizan en matrices ortogonales, donde las filas forman grupos de paridad (similar al RAID 5) y las columnas de igual manera, cualquiera de estas dos organizaciones pueden ser utilizadas para reconstruir la falla existente en un disco.

- **RAID 10.** Sistema basado en el funcionamiento de RAID 0 y RAID 1, el primero caracterizado en almacenar los bloques de información de manera distribuida y el segundo encargado de duplicar la información, en esta configuración se requiere de dos canales y dos discos por cada canal para redundancia, se utiliza la mitad de la capacidad para información de control.

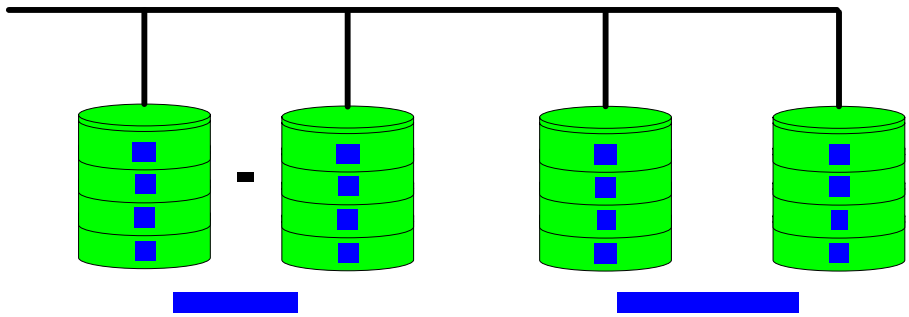


Figura 2-10. RAID nivel 10.

- **RAID 0 + 1.** Permite un acceso rápido a los datos, creado por dos líneas de RAID 0 y sobre estas un espejo de RAID 1.

- **RAID 1 + 0.** Esquema más confiable y rápido, caracterizado por tener varios espejos de RAID 1 y sobre estos se genera un nivel RAID 0, esquema que necesita muchos discos.

Estos son los niveles de RAID más importantes, pero existen otros niveles de RAID como el RAID 30 y RAID 50, los cuales son una combinación de los esquemas detallados anteriormente, y fueron creados de acuerdo a las necesidades y requerimientos que algunos sistemas requerían.

2.1.5.1.3 LVM

Logical Volume Management, en un principio fue desarrollado por IBM, posteriormente adoptado por OSF (*Open Standards Foundation*, ahora OpenGroup). Módulo[29] que se agrega al kernel de Linux con lo que se logra tener un grupo de discos y particiones como uno solo, logrando con esto dar a entender al sistema de que se trata de una sola imagen.

[29] Un módulo es una pieza de software que incluye características y funcionalidades que se desean incluir en el arranque del sistema, sin embargo que puedan retirarse o agregarse en tiempo de ejecución. La versión análoga a un módulo son las librerías de enlace dinámico o DLLs de Microsoft © Windows.

Un LVM consta de tres partes:

- **Volúmenes Físicos (FV).** Formados por los discos o particiones de estos.
- **Volúmenes Lógicos (VL).** Equivalente a una partición de un sistema tradicional. Ideal para crear sistemas de archivos.
- **Grupos de Volúmenes (GV).** Lugar donde se juntan los volúmenes lógicos y físicos.

Los volúmenes físicos como ya se dijo constituyen los discos o particiones físicas, los cuales se los pueden agrupar dependiendo de las necesidades para conformar una sola imagen, esto permite crear los volúmenes lógicos.

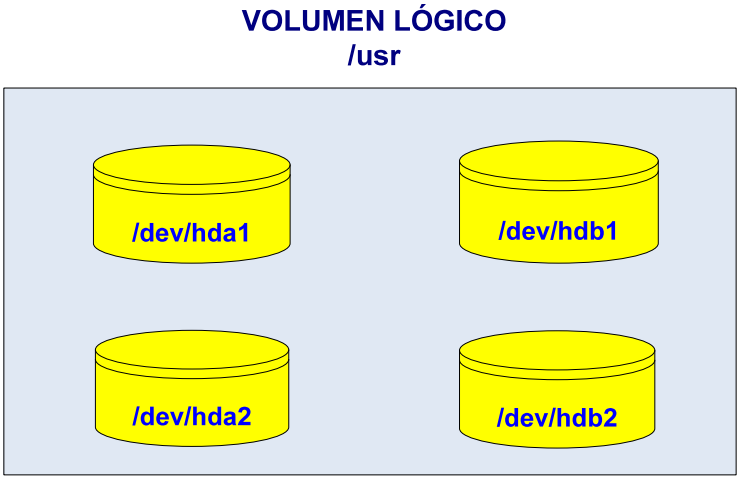


Figura 2-11. Volúmenes Lógicos

Con esta ventaja el administrador tiene la potestad de disminuir o agrandar el VL de acuerdo a su conveniencia, ya sea para reemplazar un disco con falla, cambiar un disco por otro existente en el sistema o ya sea para reemplazarlo. En un sistema puede existir varios volúmenes lógicos, siendo una de sus principales características la de que cualquier cambio que se haga en un volumen lógico, es transparente para el usuario y, el sistema continua funcionando sin ningún tipo de interrupciones. El grupo de volúmenes es el que se encarga de agrupar a todos los volúmenes lógicos y físicos.

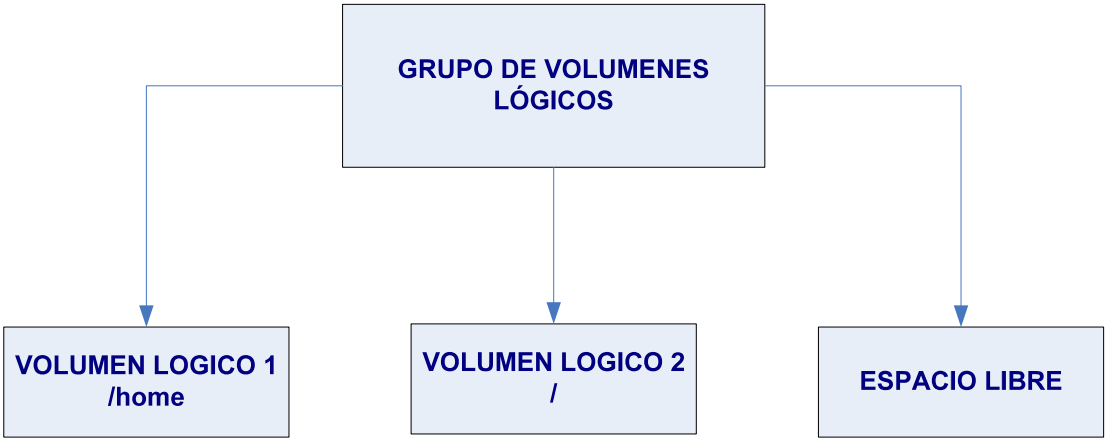


Figura 2-12. Grupo de Volúmenes Lógicos

Los volúmenes lógicos tienen la ventaja de escalabilidad, es decir que si la capacidad de un grupo lógico llega a su límite, este se lo puede ampliar agregando un disco a un determinado volumen lógico. La migración de datos de un disco a otro es otra ventaja de estos volúmenes, la cual se puede realizar en caliente.

2.2 HERRAMIENTAS PARA IMPLEMENTAR CLUSTERS DE ALTA DISPONIBILIDAD Y BALANCEO DE CARGA.

2.2.1 OSCAR HA.[13]

2.2.1.1 Introducción.

Antes de empezar a realizar un estudio de HA-OSCAR (*High Availability* - OSCAR), se debe hacer una breve reseña de donde surgió, OSCAR HA tiene sus bases en el desarrollo de OSCAR (*Open Source Cluster Application Resources*) el cual es un software de paquetes que fue diseñado por el Grupo de Clusters Abiertos (*OCG – Open Cluster Group*), grupo que con el apoyo de otros se ha dedicado a perfeccionar de mejor manera la utilidad de este paquete con la finalidad de aumentar su utilización.

El diseño global de OSCAR ha permitido la realización de un paquete compacto el cual facilita la utilización del mismo, con lo cual se busca poder instalar y configurar los paquetes necesarios que el *cluster* necesita de manera automática.

El primer desarrollo de OSCAR fue la versión 1.0, la cual ofrecía una herramienta de alto rendimiento, la cual es de mucha utilidad para ser ocupada en sistemas que requieren una gran capacidad de cálculo y procesamiento, pero esta necesidad se resuelve uniendo la potencia de cálculo de los nodos pertenecientes al *cluster*.

Con el pasar del tiempo OCG efectuó varios estudios con los que pudo desarrollar dos herramientas que son de mucha utilidad en el ambiente *clustering*. Estas herramientas son: la solución de *clusters* Thin-OSCAR, y la solución de alta disponibilidad HA-OSCAR.

Thin-OSCAR es una herramienta desarrollada por el grupo de trabajo de OSCAR que lleva su mismo nombre, con esta técnica se logró reducir el costo de implementación de los servidores debido a que se eliminaron los discos locales, por el hecho de no poseer un disco local los sistemas que funcionen con esta herramienta necesitan de un disco de arranque del sistema operativo Linux, el cual debe ser ejecutado vía red o por medio de un disco de arranque físico como un *diskete*, además, se debe mencionar que los sistemas de archivos y directorios se transfieren a los servidores vía red utilizando NFS.

Cabe recalcar que esta es una forma implementar Thin-OSCAR con nodos sin discos locales llamados *diskless*, existen otras formas de hacerlo utilizando nodos sin sistema operativo (*systemless*[30]). Y nodos con disco y sistema operativo (*diskfull*[31]*).*
La otra herramienta desarrollada por OCG es aquella que ofrece alta disponibilidad llamada HA-OSCAR, la cual fue desarrollada por el grupo de trabajo de alta disponibilidad de OSCAR.

---

[30] *systemless* - los cuales se caracterizan por tener uno o varios discos locales utilizados para almacenamiento temporal de la memoria *swap,* en estos no se almacena ningún sistema operativo.

[31] *diskfull - . Nodos con presencia de nodos locales en su sistema.*

HA-OSCAR introduce varias ventajas y nuevas características a OSCAR, principalmente ofreciendo mecanismos de alta disponibilidad para evitar puntos de fallo, escalabilidad, y de seguridad, con lo cual brinda una gran ventaja a los ambientes *clustering* tales como los de alto rendimiento, balanceo de carga; entre otros.

2.2.1.2 Arquitectura de HA-OSCAR

La arquitectura de HA-OSCAR se puede resumir en el siguiente gráfico:

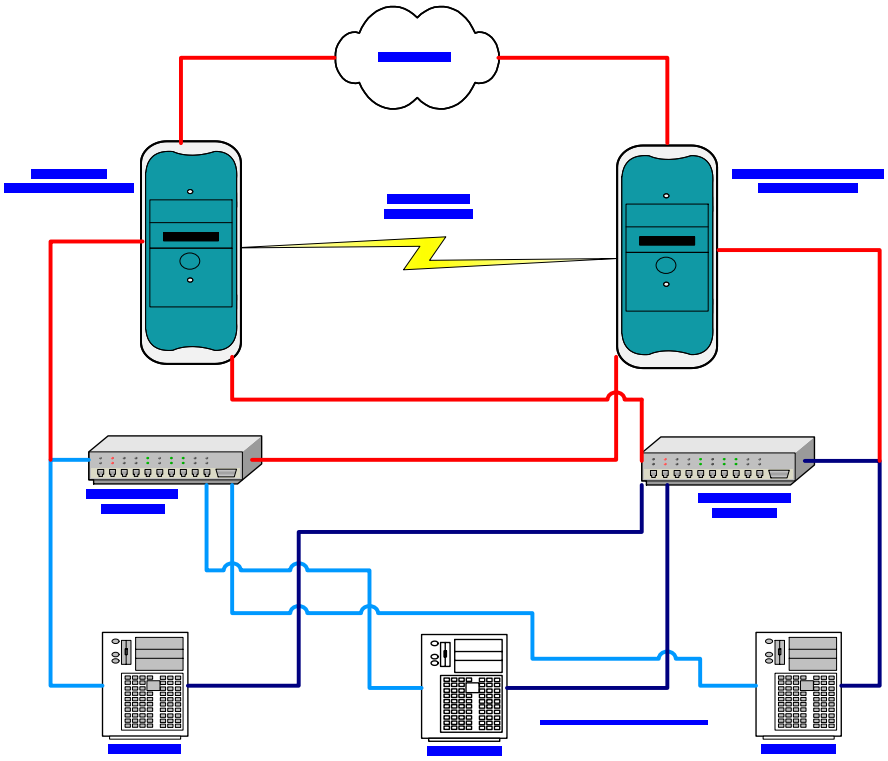


Figura 2-13. Arquitectura HA-OSCAR

Como se puede observar en la Figura 2-13, HA-OSCAR posee algunos componentes que se detallan a continuación:

- Servidor primario activo y redundante. Encargados de recibir y distribuir los requerimientos especificados por los clientes. Cada servidor posee tres

interfaces de red, una de ellas es utilizada para conectarse a Internet por la cual los clientes saldrán al mundo exterior, las otras dos interfaces son utilizadas para conectarse a la red LAN privada, estas deben conectarse a cada uno de los switchs respectivamente, esto para brindar redundancia en caso de la falla de uno de los servidores primarios, con esta configuración se elimina un punto de falla a nivel de mencionados servidores, entre éstos se tiene configurado un enlace de *heartbeat* el cual es una técnica de monitorización entre dos o más nodos de un *cluster*, es decir mediante esta herramienta se puede conocer si un determinado equipo está en funcionamiento o no. Esta funcionalidad va ayudar a determinar si un servidor primario falla y por ende empezar a funcionar el redundante quien debe suplir las funcionalidades del principal. Este servidor redundante es un servidor de imágenes el cual utiliza estas para construir y restaurar imágenes del servidor primario, así como para generar respaldos en caso de una falla inesperada.

Esta arquitectura permite tres modos de configuración entre los servidores primario activo y redundante, a continuación se da una breve explicación de estos:

*Active-Active. C*onfiguración donde los dos servidores se encuentran activos, con lo cual se puede lograr una mejor utilización de los recursos del *cluster*, si uno de ellos llega a fallar el otro toma el control del sistema existiendo una pérdida de rendimiento y no de corte de servicio.

Active-Hot Standby. Configuración donde existe un servidor principal y uno redundante, este es una fiel copia del principal y entra en funcionamiento el momento que detecta que el servidor principal tuvo una falla, el servidor redundante utiliza imágenes del sistema para construir y restaurar el sistema del servidor primario.

Active-Cold Standby. Configuración similar a la *Active-Hot Standby,* con la diferencia que el servidor redundante al momento en el que detecta que el principal ha fallado, entra en funcionamiento sin ningún conocimiento del trabajo que estaba cumpliendo el principal.

- Clientes. Quienes generan peticiones y requerimientos de cómputo a los servidores primarios.

- Switches. Equipos de interconectividad encargados de comunicar a clientes y servidores, es importante que exista redundancia en estos equipos para garantizar una alta disponibilidad en todos los componentes que conforman el *cluster* y con ello evitar la existencia de puntos de fallo.

2.2.2 LVS (LINUX VIRTUAL SERVER).[14]

2.2.2.1 Introducción a LVS

A continuación se enumeran los principales requerimientos que debe tener un sistema de alta disponibilidad para hacer frente a la inminente demanda de conexiones y servicios en Internet:

- **Escalabilidad.** Para cuando la demanda de un determinado servicio aumenta, el sistema debe ofrecer la capacidad de ampliarse con la finalidad de atender todas las solicitudes sin que los tiempos de respuesta sean alterados.

- **Alta Disponibilidad.** Capacidad del sistema de estar disponible en todo momento, tomando en cuenta de que debe ser un sistema altamente tolerante a fallos.

- **Costos.** Uno de los requerimientos fundamentales a la hora de hacer una inversión, los costos de montaje, mantenimiento y expansión del sistema deben ser accesibles en comparación a infraestructuras propietarias.

El proyecto de Servidor Virtual en Linux (LVS – *Linux Virtual Server*), fue iniciado por Wensong Zhang. Su objetivo principal es el de utilizar una red de computadoras u otros dispositivos para que sean accedidos desde el exterior dando la imagen de un recurso único. Para ello requiere de varios servidores interconectados entre sí, donde exista uno que cumpla las funciones de director (nodo director) o servidor virtual y sea el encargado de suministrar las funciones a numerosos equipos de pequeño tamaño

y costo reducido, para que presten un determinado servicio, además coordina que cada equipo colabore para brindar una alta capacidad computacional que crece casi linealmente respecto a la capacidad computacional que ofrece cada equipo de forma individual.

El proyecto Linux *Virtual Server* (LVS) ofrece parches y aplicaciones de mantenimiento y gestión que permiten construir un *cluster* de servidores que implementa alta disponibilidad y balanceo de carga sobre el sistema operativo GNU/Linux.

2.2.2.2 Componentes de LVS

El proyecto LVS posee varios componentes. Aquel de mayor importancia lo conforman los parches del *kernel*, los cuales se aplican al código fuente del núcleo del sistema operativo Linux y que una vez compilado el núcleo ofrece las nuevas funcionalidades. Para esto el nuevo código implementado interactúa con las rutinas de manejo del protocolo TCP/IP. La última versión de LVS para sistemas Linux viene incorporado en el *kernel* (versión 2.6.x), y no es necesario instalar parches o recompilar *kernel* para disponer de sus características, salvo casos especiales donde sea necesario. Se requiere incluir herramientas de usuario útiles para la administración y gestión del sistema.

La herramienta *ipvsadmin* permite al usuario configurar al servidor virtual, esta herramienta es de mucha utilidad para modificar los parámetros de funcionamiento del servidor virtual, modificando los parámetros de *kernel* del sistemas operativos Linux, esta modificación puede realizarse mediante el interprete de comandos o basado en ordenes dentro de un *script* de *shell*. La existencia de otras herramientas como son los *scripts ipvsadm-save* e *ipsadm-restore* también son importantes ya que permiten guardar los parámetros actuales del sistema en un archivo y restaurarlos respectivamente.

2.2.2.3 Arquitectura

La arquitectura del *cluster* de servidores virtuales está conformado de tres componentes fundamentales como se muestra en la Figura 2-14.

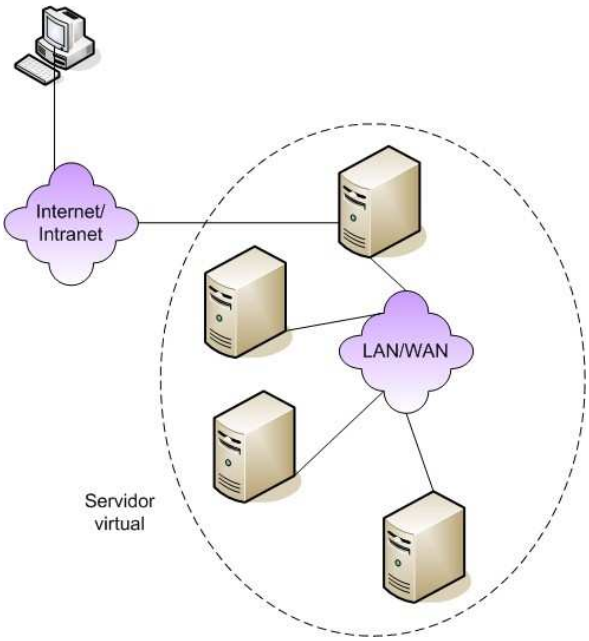


Figura 2-14. Arquitectura de LVS

- **Servidor Virtual.** Servidor que da la cara al exterior, es decir quien recibe en primer plano los requerimientos del cliente. Éste se caracteriza por tener configurada una dirección IP pública mediante la cual se publica el servicio ofrecido. Este servidor se encarga de recibir múltiples conexiones y redirigirlas a los servidores reales mediante un proceso de balanceo de carga.

  La aplicación de servidor no se ejecuta en este equipo y el software encargado de redireccionar las peticiones al cliente se ejecuta el *kernel* del sistema operativo, evitando con esto una sobrecarga de procesamiento con relación a si ejecutase mencionada aplicación.

- **Servidores Reales.** Servidores que se encuentran detrás del servidor virtual y cuya principal función es la de ejecutar el servicio que se está ofreciendo. Esta aplicación puede ser cualquiera que utilice el conjunto de protocolos TCP/IP.

- **Sistemas de Archivos Distribuidos.** Sistema que no es un componente fundamental, este permite centralizar la información y con ello tener una mejor administración de los datos, aunque los servidores reales pueden tener su acceso local a los mismos y operar normalmente.

Como se puede apreciar en la Figura 2-14, el sistema da la cara al usuario mediante un único servidor virtual conocido también como nodo director, éste se encarga del balanceo de carga y reenvía las peticiones a los servidores reales que conforman el *cluster*. El nodo director ejecuta el sistema operativo Linux con un *kernel* parchado para incluir el código IPVS (*Internet Protocol Virtual Server*), el cual permite realizar el balanceo de carga vía software, este código puede instalarse como una herramienta adicional mediante línea de comandos.

El nodo director hace las funciones de ruteador, el cual basado en reglas de enrutamiento redirecciona el tráfico entrante a los servidores reales. El momento que la comunicación se establece esta perdura durante el periodo que la conexión TCP se utiliza y cuando se inicia requiere de una nueva conexión el director selecciona un nuevo servidor real diferente al anterior.

Con esta arquitectura, además del balanceo de carga, estamos permitiendo que los servidores reales individuales puedan ser extraídos del LVS, actualizados o reparados y devueltos al *cluster* sin interrumpir la prestación de servicio. Asimismo, el sistema es fácilmente escalable. Para brindar alta disponibilidad en LVS se utiliza enlaces de respaldo o de latido de corazón (*heartbeat*) entre servidores tipo espejo, sistemas de archivos distribuidos y dispositivos redundantes.

2.2.2.4 Mecanismos de Balanceo de Carga

2.2.2.4.1 Balanceo por NAT

Balanceo de carga caracterizado por aprovechar la funcionalidad que presenta el *kernel* de Linux para hacer que el nodo director tenga un comportamiento de enrutador con NAT (*Network Address Traslation*); propiamente el nodo director utiliza una extensión del NAT conocida como Traducción de Direcciones de Red y Puerto o NAPT (*Network Address Port Translación*), esta se caracteriza por usar una sola dirección IP válida con la que los host se conectan desde la red interna hacia el Internet por diferentes puertos. NAPT se basa en el mapeo de direcciones de red y puertos TCP/UDP de los paquetes en otros puertos. NAPT es una técnica que permite que numerosos servicios se ejecuten en distintas direcciones de red y se

escuchen en distintos puertos y, a su vez puedan ser trasladados a un único puerto en la dirección de red del servidor virtual para lograr la apariencia de un único servicio.

El funcionamiento de esta técnica de balanceo de carga se resume en la Figura 2-15.

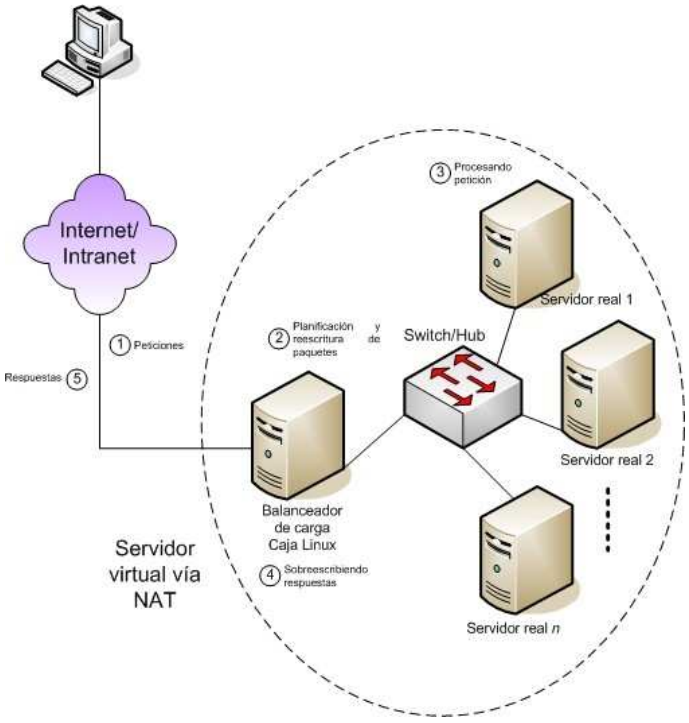


Figura 2-15. Balanceo por NAT

El proceso de balanceo de carga se resume en los siguientes pasos:

1. El cliente genera un requerimiento al nodo director el cual posee una dirección IP pública y es el que da la cara al mundo exterior.

2. El nodo director establece a que servidor real enviar la petición requerida por el cliente utilizando NAPT y reescribe las cabeceras del paquete TCP/IP, modifica los campos de dirección IP y puertos de destino, los suplanta por los correspondientes al servidor real escogido. Cabe recalcar que los servidores reales y el nodo director se encuentran interconectados por una red de alta velocidad mediante un *switch* o *hub*.

3. El servidor real escogido recibe la petición, la procesa y envía la respuesta al cliente, pero como en esta configuración los servidores reales tienen como ruta de salida (*default gateway*) al balanceador de carga, la respuesta se envía a través de éste. El nodo director suplanta nuevamente la dirección IP y puerto de origen de los paquetes TCP/IP con la correspondiente IP pública con la que se generó la petición, y de esta manera la respuesta llega al cliente final.

La utilización de esta configuración no requiere de ninguna modificación especial del sistema operativo del cliente, lo cual facilita su uso.

El número de servidores reales hasta el que el *cluster* pueda escalar dependerá del ancho de banda de las conexiones con el balanceador y, de su potencia de cálculo para reescribir los paquetes TCP/IP. De todas formas, un servidor actual no debería tener problemas para tratar con 20 o tal vez más servidores.

La principal desventaja del balanceo por NAT está en la sobrecarga que tendría el nodo director al tener que reescribir todos los paquetes de datos originados desde y hacia los servidores reales y clientes, problema que se hace notorio cuando el número de peticiones y respuestas es muy grande. Este problema puede disminuirse utilizando un equipo más potente como director. Otra solución a este problema consiste en que los servidores reales envíen los paquetes de datos de salida directamente hacia los clientes sin pasar por el nodo director mediante las técnicas que se describen a continuación.

2.2.2.4.2 Balanceo por Encapsulado IP.

Técnica caracterizada por encapsular un datagrama IP dentro de otro, es decir el encapsulado IP consiste en enmascarar una trama TCP/IP, con sus direcciones IP de origen y destino, dentro de otra trama con direcciones distintas, una vez que la trama más externa llegue a su destino, desencapsular la trama original y reenrutarla desde allí.

Como requisito principal esta técnica requiere que todos los dispositivos soporten el protocolo IP *tunneling* o IPSec.

El funcionamiento de esta técnica de balanceo de carga se resume en la Figura 2-16.

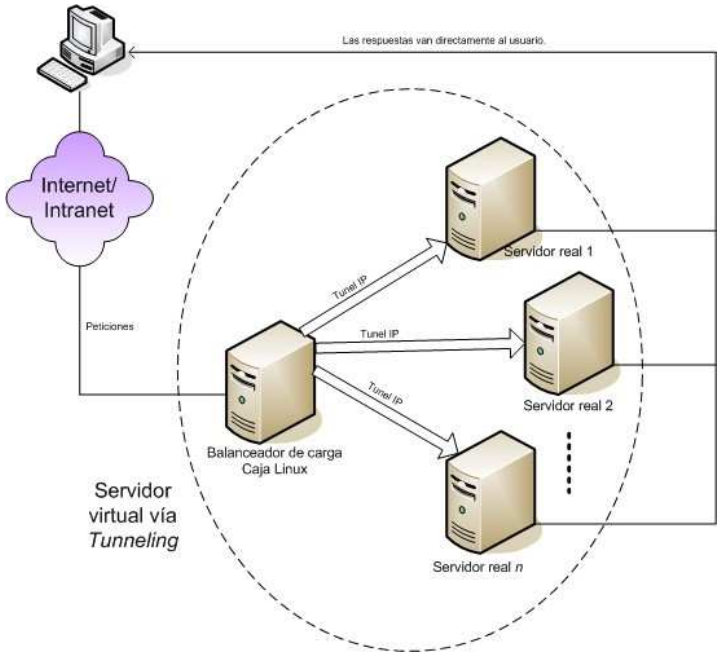


Figura 2-16. Balanceo por encapsulado IP

Descripción del proceso:

1. El cliente envía sus requerimientos a la dirección IP pública o virtual VIP, la cual está configurada en el nodo director o balanceador.

2. El balanceador de carga o nodo director recibe las peticiones del cliente, y realiza el proceso de encapsulación, es decir toma los datagramas IP del cliente y los coloca dentro de otros datagramas IP que son reenviados hacia los servidores reales, para el reenvío de los paquetes, el nodo director encapsula los datagramas provenientes del cliente los cuales poseen su dirección IP como origen y destino en la dirección IP del balanceador, este procede a encapsular este datagrama dentro de otro con su dirección IP como origen y la dirección IP del servidor real como destino.

   El trayecto de llegada de los paquetes a los servidores reales puede darse de manera directa cuando están en una misma área o por medio de dispositivos y redes IP intermedias cuando se encuentren en áreas diferentes.

3. Una vez que el paquete llega al servidor real, este debe proceder a desencapsularlo, para ello todos los servidores reales deben tener configurados en una de sus interfaces la IP pública para luego de procesar los requerimientos del cliente responderle de manera directa sin necesidad de hacerlo mediante el nodo director o balanceador, es decir los servidores reales utilizan la ip real como origen y la del cliente como destino. 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, mientras que los de salida los enviará cada servidor real directamente a su destino.

Este método de balanceo de carga posee problemas debido a que tanto el nodo director como los servidores reales poseen en una de sus interfaces la misma IP pública configurada. En configuraciones donde los servidores reales y el nodo director están en la misma red, existen inconvenientes en el momento en que llega una petición a la dirección IP pública del sistema. Los servidores reales y el director responderían a requerimientos ARP (*Address Resolution Protocol*), creando un conflicto interno y haciendo que los paquetes sean enviados tanto al director como a los servidores reales, difiriendo el propósito del mismo, el propósito de éste método se refiere a que al nodo director o balanceador le lleguen todas las peticiones y sea este quien responda a los requerimientos ARP y redireccionarlas a los servidores reales. Para ello se debe lograr que los servidores reales que se encuentran en la misma red del nodo director no respondan a requerimientos ARP y, procesar los paquetes destinados a la IP pública de manera local, ello se logra ocultando las interfaces donde se tiene configurada la IP pública, proceso que se detallará en el próximo capítulo.

La tecnología de balanceo de carga por IP *tunneling* posee una escalabilidad mayor que la que presenta el balanceo basado por NAT, éste permitirá escalar hasta un mayor número de servidores, 100 o más, con la condición de que todos soporten encapsulado IP (IP *tunneling*).

Una de las ventajas de esta tecnología es que posibilita distribuir los servidores reales a lo largo de una red de área amplia en lugar de tenerlos todos en un mismo segmento de red local.

2.2.2.4.3 Balanceo por enrutamiento directo

Mecanismo de balanceo de carga, caracterizado por utilizar una red LAN con sus respectivos dispositivos de interconexión como un *Switch*/*Hub* para la interconexión entre el nodo director y los servidores reales,

La característica de este método radica en que los servidores reales poseen configurado en el interfaz de red local, la dirección IP real del nodo director, pero como en el caso de balanceo por encapsulamiento IP deben estar configuradas para que no respondan a mensajes del protocolo ARP.

El funcionamiento de este método se resume en la Figura 2-17.

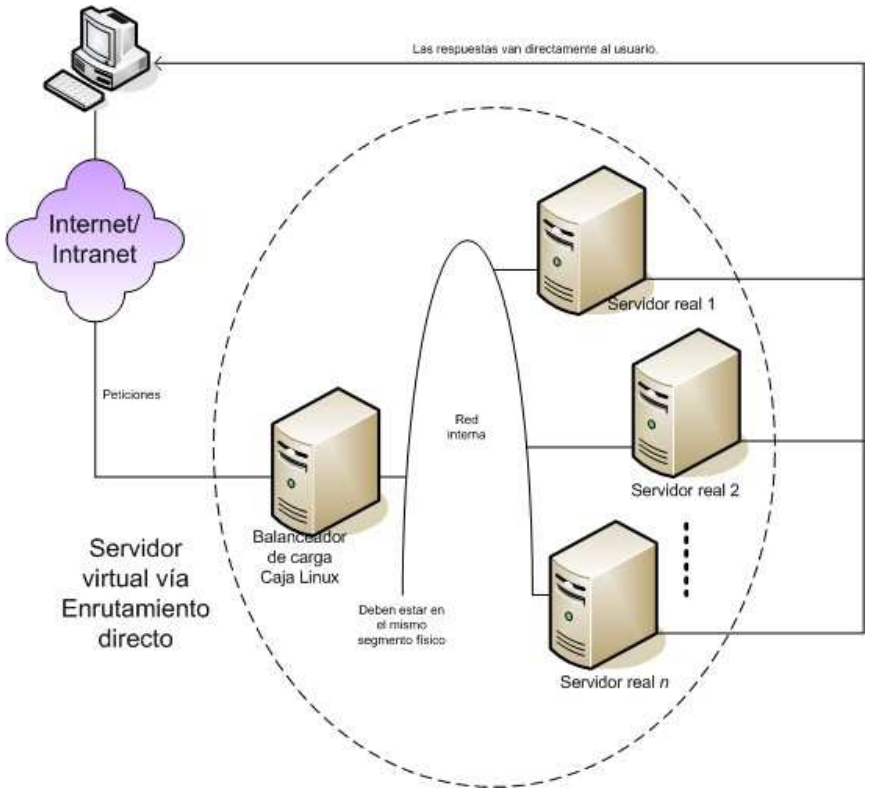


Figura 2-17. Balanceo por enrutamiento directo

Descripción de la figura:

1. Como se puede observar el nodo director es quien recibe las peticiones desde los clientes que están en el Internet hacia la dirección IP pública configurada en una de sus interfaces.

2. Todos los equipos tienen configurado en una de sus interfaces la IP pública del *cluster* y, tomando en cuenta que el balanceador siempre la usará para el acceso a Internet y, será el punto de entrada al *cluster*. El resto de equipos se conectan al balanceador en la misma red física y en el interfaz conectado a esta red tendrá configurada la IP pública del *cluster*, el cual no responde a comandos ARP (todos los equipos responderían por la misma IP con distintas MACs). Cuando llega una petición al balanceador éste decide a qué servidor enviársela, y redirige el paquete a nivel de capa enlace a la dirección MAC del servidor real elegido, en lugar de modificar o encapsular el paquete TCP/IP.

3. Una vez recibido el paquete TCP/IP enviado por el nodo director hacia el servidor real elegido, este lo analiza hasta el nivel de red, acepta el paquete y genera la respuesta que enviará directamente al cliente sin necesidad de reenviársela al nodo director, evitando la sobrecarga que se tenía en la técnica de encapsulado.

Este método no es aplicable para todos los interfaces, esto se debe a que varios fabricantes de interfaces de red no proveen soporte para deshabilitar el protocolo ARP.

Al realizar el reenrutamiento a nivel de capa enlace, se imposibilita tener el *cluster* disperso en áreas geográficas extensas, y requiere que los componentes del *cluster* se encuentren el mismo segmento físico.

2.2.2.5 Algoritmos de Balanceo de Carga.

2.2.2.5.1 Round Robin.

También llamado FIFO (*First Input First Output*), proceso por el cual el nodo director envía una petición al primer servidor del *cluster,* la siguiente petición se envía al

siguiente servidor disponible y así sucesivamente, hasta llegar al último servidor disponible, luego del cual se vuelve a enviar al primero. Este mecanismo es el más sencillo de todos pero no el más justo ya que se puede dar el caso en el que toda la carga pesada vaya a un determinado servidor, mientras que el resto recibe peticiones más sencillas.

2.2.2.5.2 Round Robin ponderado.

Algoritmo cuyo funcionamiento es igual al Round Robin, con la diferencia que a cada servidor se le da un indicativo de acuerdo a su potencia de cálculo; es decir servidores con alta potencia de cálculo recibirán mayor carga que los que poseen una potencia menor. Pero este método también tiene la desventaja de que los servidores más capaces reciban más carga y con ello llegar a saturarlos.

2.2.2.5.3 Servidor con menos conexiones activas.

Proceso caracterizado por tener un mecanismo continuo de consulta acerca de cuántas conexiones activas poseen los servidores que están atendiendo los requerimientos del cliente, dependiendo de esta consulta el nodo director direcciona las peticiones a los servidores con menos cargas. Pero este proceso es muy útil cuando todos los servidores poseen una potencia de cálculo similar, logrando tener de esta manera un trabajo equilibrado. El inconveniente surge cuando la potencia de los servidores no es la misma en todos, en este ámbito los servidores más rápidos recibirán un mayor número de conexiones activas, así como conexiones en espera, los servidores más lentos tendrán un número menor de conexiones activas y en espera, lo cual ocasiona que el nodo director envíe más carga a estos servidores por el hecho de tener pocas conexiones y con ello se logra que éstos lleguen a saturarse.

2.2.2.5.4 Servidor con menos conexiones activas (ponderado)

Algoritmo que basa su funcionamiento de acuerdo al menor número de conexiones activas y de un determinado indicativo que se le da a los servidores en base a su capacidad de cálculo.

Este proceso consiste en asignar una mayor carga a los servidores con menos conexiones activas pero verificando la capacidad de cálculo de éstos, y dependiendo de esta capacidad asignar una mayor o menor carga a los servidores.

2.2.2.5.5 Menos conectado basado en servicio

Algoritmo caracterizado por dirigir todas las peticiones a un mismo servidor hasta lograr que se sobrecargue, es decir que tenga un gran número de conexiones activas que superen a su potencia de cálculo, luego de esto se ejecuta el algoritmo de menos conexiones activas ponderadas sobre el resto de servidores del *cluster*, con ello se pasan las peticiones al servidor con menos conexiones activas analizando previamente la potencia de cálculo, el proceso continua hasta que se satura la capacidad de conexiones activas. Este mecanismo se mantiene de manera sucesiva hasta llegar al último servidor en servicio del *cluster.* Este algoritmo es muy utilizado cuando se ofrece varios servicios distintos y se quiere especializar a una determinada máquina en un servicio determinado, cabe recalcar que todas las máquinas son capaces de reemplazar a cualquiera de lo servidores activos cuando uno de estos falle.

2.2.2.5.6 Tablas hash por origen y destino

Métodos que dispone de una tabla de asignaciones fijas, en las que por medio de la dirección IP de origen o de destino, se indica qué servidor deberá atender la petición. El balanceador compara las direcciones de las tramas TCP/IP que reciba con estas tablas y actúa en consecuencia.

2.2.2.5.7 Conexiones persistentes

Algoritmo muy útil, que basándose en los ya analizados se puede añadir que, una vez que un cliente se haya conectado a un determinado servidor siempre el nodo director le dirija al mismo, logrando con esto tener conexiones persistentes útiles cuando se requiere mantener conexiones activas a una determinada aplicación.

2.2.2.6 Alta disponibilidad en LVS[15]

2.2.2.6.1 Heartbeat

.

Herramienta que ofrece alta disponibilidad y basa su funcionamiento de acuerdo a la combinación de varias herramientas agrupadas de la siguiente manera:

- **mon+heartbeat+fake+coda.-** Alta disponibilidad producida por la combinación de varios productos basados en software, estos son:

  - **mon.** Recurso útil para monitorizar los servicios del sistema tanto en los nodos directores principal y de respaldo, así como en los servidores reales, esto con la finalidad de brindar alta disponibilidad de los servicios todo el tiempo.

  - **Heartbeat.** Técnica de monitorización entre dos o más nodos de un *cluster*, es decir mediante esta herramienta se puede conocer si un determinado equipo está en funcionamiento o no.

  - **Fake.** Mecanismo de suplantación de identidad que se activa el momento que un equipo a caído, es decir por medio de fake, se puede suplantar la identidad de otro tomando su dirección IP y evitar con esto poseer puntos de falla dentro de un sistema.

  - **Coda.** Sistema de archivos distribuido redundante, tolerante a fallas.

La Figura 2-18 mostrada a continuación, resume el proceso de alta disponibilidad que ofrece los mecanismos mon+heartbeat+fake+coda:

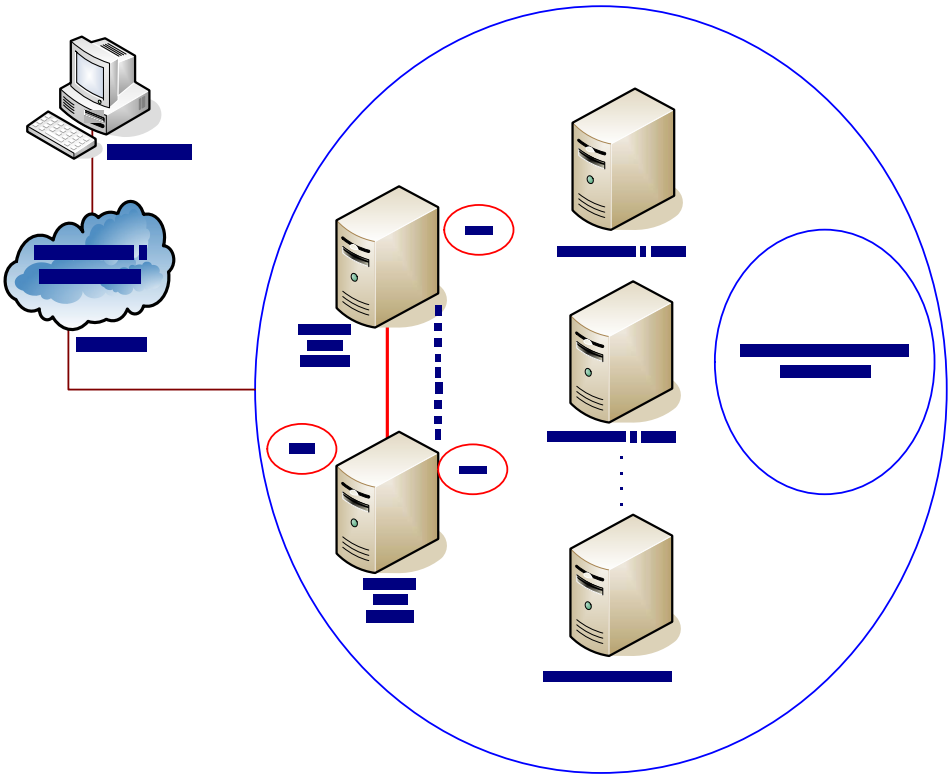


Figura 2-18. *Heartbeat*

El proceso de estos mecanismos tiene tres etapas y se resume de la siguiente manera:

- **Balanceador de Carga.-** Como se vio en anteriores ocasiones, es quien da la cara al mundo exterior y quien recibe las peticiones del cliente, su funcionalidad es de vital importancia, ya que si queda inoperante, el *cluster* quedaría fuera de servicio, es por ello que para evitar tener un punto de fallo, este balanceador debe tener un respaldo, el cual debe tener la misma configuración del balanceador principal y con ello garantizar alta disponibilidad. Estos dos balanceadores (nodos directores) deben monitorizar su funcionamiento de manera dual mediante el uso de *mon* para ver el estado de los servicios y usan la herramienta *heartbeat* mediante mensajes UDP heartbeat para verificar el funcionamiento de ambos, en caso de que exista un fallo en el nodo principal se utilizará *fake* para suplantar su identidad, es decir

toma el nodo de respaldo la dirección IP principal y con ello el control de todo el *cluster.*

- **Servidores Reales.-** La segunda etapa es la de los servidores reales, la alta disponibilidad se logra mediante el monitoreo que hace el nodo director o balanceador utilizando *mon,* este proceso se realiza *a* nivel de máquina utilizando el demonio fping.monitor y a nivel de servicio utilizando el demonio que depende del servicio que se esté monitoreando, como ejemplo http.monitor para ver el estado del servicio http, ftp.monitor, para ver el estado de el servicio ftp; y así con el resto de servicios que el balanceador o nodo director está monitoreando. En caso de que los servicios de los servidores reales no den respuesta al censado que el nodo director realiza, se genera una alarma que está previamente programada y cuya función es notificar al nodo director de que un determinado servidor no está operando adecuadamente y con ello debe añadir una regla en las tablas de LVS indicando que mencionado servidor esta fuera de servicio y que no se debe redireccionar las peticiones del cliente hasta el momento en que se encuentre operativo nuevamente y sea reingresado al *cluster.*

- **Servidores de Archivo.-** La tercera etapa consiste en tener un sistema de ficheros distribuido CODA, el cual puede estar montado en varios servidores con la finalidad de tener alta disponibilidad ante cualquier fallo que pudiera ocurrir.

Una consideración importante que hay que tener en cuenta, es que se debe proporcionar una alta disponibilidad en los equipos de interconectividad como *routers* y *switch* ya que si estos son un punto de fallo dejarían fuera todo el cluster y con ello la disponibilidad de este.

- **ldirectord+heartbeat.-** Alta disponibilidad similar a la vista en el apartado anterior, la diferencia radica en que la monitorización se realiza a través de ldirectord y no por medio de *mon*, la función de este consiste en:

- **ldirectord (*Linux Director Daemon*).-** Demonio escrito por Jacob Rief, las funciones de *ldirectord* son similares a las que realiza *mon*, con la diferencia de que únicamente permite monitorear servicios http y https y no cualquier servicio como si lo permite *mon*. Este demonio fue creado específicamente para ser usado por LVS y por ello es fácil de instalar e integrar en un *cluster* de este tipo. El funcionamiento de *ldirectord* se caracteriza por que este demonio lee directamente los archivos de configuración de LVS, y por ende modificar las tablas de enrutamiento IPVS en caso de que exista algún problema, eliminando el servidor real que posea inconvenientes y evitar reenviar a éste peticiones del cliente.

  ldirectord tiene un buen funcionamiento con heartbeat, ya que puede ser iniciado fácilmente cuando el caso amerite.

En la Figura 2-19 mostrada a continuación se resume el funcionamiento de este método de alta disponibilidad

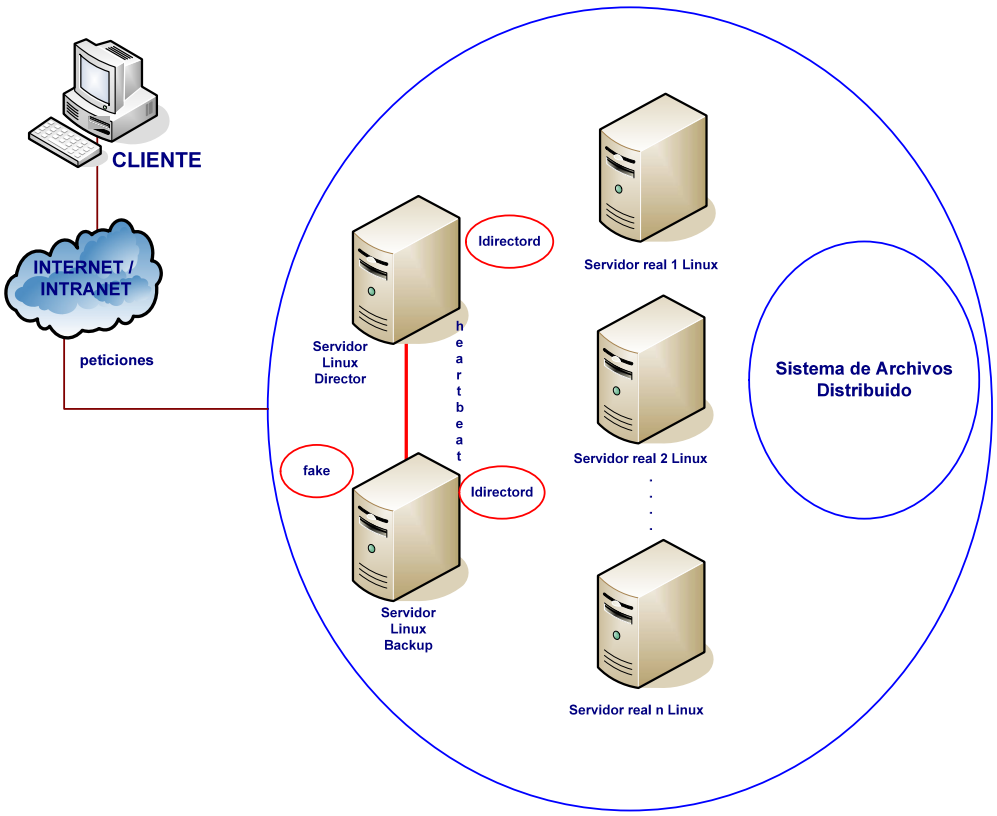


Figura 2-19. *ldirectord+heartbeat*

Como se puede observar la diferencia con el otro mecanismo de alta disponibilidad radica en la monitorización de servicios, en este caso se realiza con ldirectord, el resto de demonios como fake, heartbeat cumplen la misma función que se especifico en el apartado anterior, de igual manera se puede tener un sistema de archivos CODA distribuido tolerante a fallas para garantizar alta disponibilidad en los servidores de archivos.

2.2.2.7 Otras herramientas de instalación y configuración basadas en LVS.

2.2.2.7.1 Ipvsadm

Herramienta de mucha utilidad para realizar la configuración de servidores virtuales y nodo director, con *ipvsadm* se modifican los parámetros de funcionamiento del software ejecutado en el *kernel* de los servidores y que se lo realiza mediante el intérprete de comandos o utilizando un *script* en el que se detallan un conjunto de órdenes a ejecutarse.

La herramienta *ipvsadm* permitirá especificar las siguientes actividades útiles para la configuración de servidores:

- Añadir servicios y servidores.
- Quitar servicios.
- Mecanismos de planificación
- Asignación de pesos a los servidores.
- Especificar servicios y servidores los cuales serán gestionados por el servidor.

Para realizar la configuración del sistema, mediante la utilización de *scripts* permite realizarlo. Este *script* obtiene toda información necesaria para realizar la configuración del *cluster,* provee mecanismos para la inicialización de LVS, en el cual se configura las IP de las máquinas que pasan por el nodo director, también incluye los programas y archivos de configuración necesarios para especificar

parámetros de funcionamiento del *cluster* tales como mecanismos de balanceo de carga, técnicas de distribución de carga; entre otras . Este archivo sirve de mucha ayuda para no tener que realizar la configuración de manera manual.

El utilizar el *script configure* va facilitar el proceso de configuración de los servidores virtuales para LVS, pero previamente se debe tener los módulos necesarios para poder ejecutarlo.

Otra manera de realizar la configuración mediante *ipvsadm* es por medio de comandos, los cuales para poder ejecutarse requieren de un conjunto de parámetros y datos específicos.

2.2.2.7.2 Piranha

Paquete cuyo propietario es Red Hat. Inc, el cual incluye un conjunto de herramientas útiles para implementar un *cluster* de alta disponibilidad y balanceo de carga. Basa su funcionamiento en LVS y otros programas de propiedad de Red Hat. Piranha ofrece las siguientes herramientas:

El parche *IPVS* para el kernel de Linux, herramienta de mucha utilidad para hacer el balanceo de carga. Cabe mencionar que para esto piranha tiene la capacidad de añadir pesos a los servidores virtuales, los cuales se asignan basándose en la capacidad de cálculo de cada servidor real.

El demonio *lvs,* quien controla las tablas IPVS, las cuales se las puede administrar bajo la herramienta *ipvsadm*, herramienta web que permite la administración remota de sistemas y servidores.

El demonio *nanny* para monitorizar todos los servicios y servidores reales que pertenecen al *cluster*, este proceso de monitorización se debe realizar de la siguiente manera: primeramente se debe verificar si están operativas las tarjetas de red y del segmento de red, luego de ello el demonio trata de verificar conectividad con el servidor que esta siendo monitoreado; luego de constatar la actividad del servidor real, se envía una petición sencilla al puerto del protocolo que se monitorea y este debe responder de manera correcta. En caso de que un servidor no responda o su

respuesta no sea correcta durante varios censados, el demonio *nanny* lo elimina de las tablas de IPVS para así quitar al servidor caído del *cluster* y evitar enviarle requerimientos por parte del cliente. Este equipo con problemas continua censándose de manera periódica, de manera que el momento en que se encuentra completamente operativo vuelva a ser reinsertado al *cluster* de manera automática.

El demonio *pulse* encargado de verificar el estado de todos los demonios del cluster y la puesta en funcionamiento del nodo director o balanceador de carga de respaldo en caso de fallo del primario. Este proceso consiste en utilizar mensajes *hearbeat* de tipo UDP entre el nodo director y el nodo de respaldo, cabe recalcar que los dos deben mantener la misma configuración, si el nodo principal falla, el secundario suplanta su dirección IP y toma su lugar. En caso de reestablecerse el nodo que ha fallado, este detecta que han tomado su lugar y adopta el comportamiento de nodo de respaldo hasta que el nodo principal que tomo su lugar falle, luego de esto vuelve a realizarse el mismo proceso.

Otro componente de mucha importancia es la interfaz gráfica *piranha* la cual permite administrar el cluster.

2.2.2.7.3 UltraMonkey

Es una herramienta libre muy útil, rápida y fácil de instalar, muy utilizada para montar *cluster* de servidores web, es una herramienta completa ya que trae en su propio *kernel* los parches para *mon, heartbeat, fake*, parches que como se indicó en los puntos anteriores periten tener un sistema de alta disponibilidad en un *cluster* de servidores. También para garantizar el balanceo de carga utiliza la herramienta proveniente de LVS llamada *Ipvsadm.*

Ultra Money permite configurar una alta disponibilidad para un *cluster* en diferentes niveles:

- **Alta Disponibilidad simple.-** Sistema que carece de un nodo director o balanceador, la alta disponibilidad se da únicamente en base al monitoreo realizado por *mon* y *heartbeat.*

- **Alta Disponibilidad simple con balanceo de carga.-** La alta disponibilidad se la garantiza con *ldirectord* y *heartbeat* y el balanceo de carga se lo realiza basado en *Ipvsadm* añadiendo un nodo director o balanceador, el mismo que se presenta al usuario y redirecciona las peticiones a los servidores reales utilizando el método de servidores virtuales basados en NAT.

- **Alta disponibilidad con balanceo de carga.-** Funcionamiento similar al ítem anterior, con la novedad de que el nodo director o balanceador tiene un equipo de respaldo el cual entra en funcionamiento en caso de que el principal falle, este proceso lo realiza utilizando el demonio *fake* con lo cual puede suplantar su identidad.

- **Balanceo de carga, alta disponibilidad y alta capacidad.-** El balanceo de carga y la alta disponibilidad se garantizan con el ítem anterior, la alta capacidad se logra cuando los servidores reales reciben las peticiones del balanceador y las respuestas la envían directamente al cliente sin necesidad de hacerlo por medio del balanceador, esto conlleva a optimizar el ancho de banda y a evitar cuellos de botella en el nodo director o balanceador, esto se logra utilizando técnicas de servidores virtudes basados en tunneling o basados en enrutamiento directo.

Como se puede observar, esta herramienta es completa, ya que trae herramientas para garantizar balanceo de carga y alta disponibilidad, pero la única desventaja esta en que no posee una herramienta gráfica para administrar y configurar el cluster.

2.2.3 OTRAS HERRAMIENTAS. [16][17][18]

2.2.3.1 SuperSparrow

SuperSparrow es un software que permite tener un *cluster* de *clusters* con lo cual se puede lograr un sistema de balanceo de carga entre sitios geográficamente dispersos.

Esta herramienta es muy útil para el proceso de elegir el espejo de una determinada Web más próxima a la localización en la que un determinado usuario se encuentre. Para ello SuperSparrow se basa en el protocolo BGP[32] v4 (*Border Gateway Protocol version 4*).

El funcionamiento de supersparrow se resume en la Figura 2-20:

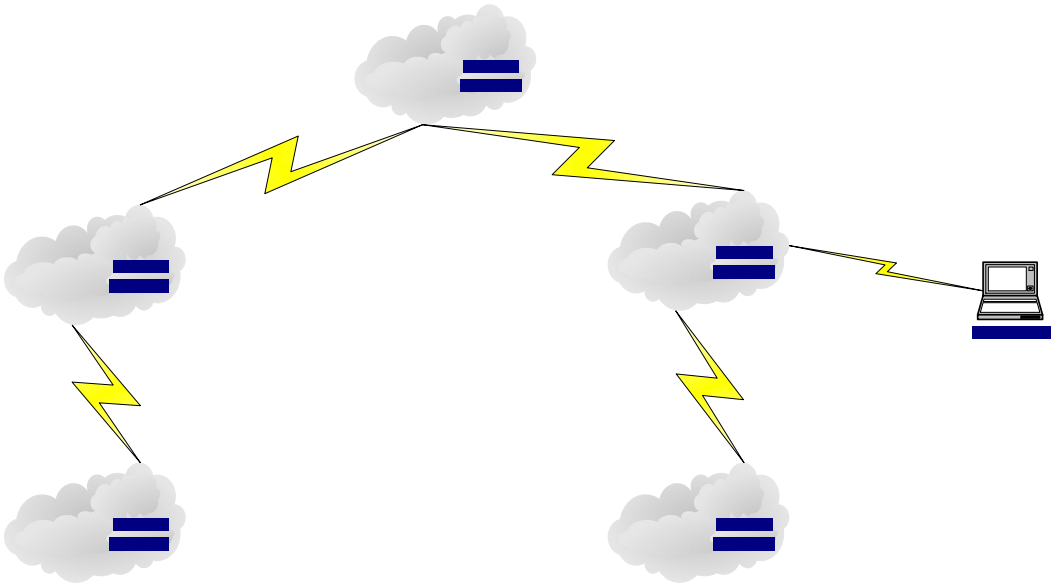


Figura 2-20 *SuperSparrow*

El gráfico presentado se tiene dos puntos de presencia POP (Points Of Presence) A y B los cuales representan la infraestructura con la cual se está prestando algún servicio (servidores web; entre otras), el momento que un cliente se conecta con cualquier POP, supersparrow mediante BGP conoce el camino AS[33] (Sistema Autónomo) hasta el cliente.

---

[32] *BGP* o *Border Gateway Protocol* es un protocolo mediante el cual los ISP registrados en Internet intercambian información de enrutamiento, es decir la totalidad de los ISP intercambian sus tablas de rutas a través del protocolo BGP. Este protocolo requiere un router BGP el cual da a conocer sus direcciones IP a los routers BGP y esta información se difunde por los routers BGP cercanos y no tan cercanos.

[33] AS (*System Autonomous)* conjunto de routers con una misma política de encaminamiento dentro de un único dominio administrativo; un AS se identifica con un número de 16 bits, es decir se pueden tener 65535 AS's

Suponiendo que un cliente se conecta a la red C cuyo AS es 003, el camino desde POP A sería AS 101 001 002 003, mientras que desde POP B sería AS 102 003. El camino más corto es el del POPB y es el preferido, por tanto supersparrow redirige la conexión del cliente hacia éste.

2.2.3.2 MOSIX

El proyecto *Mosix* es un producto comercial desde el año 2001, pero a la vez surgió una herramienta libre denominada *OpenMosix* la cual entro a operar desde el mes de febrero del 2002. Estas tecnologías son basadas en Linux y su principal función es de realizar el balanceo de carga de procesos (migración de procesos) ejecutándose en el *cluster*, esta migración se realiza de acuerdo a la velocidad del CPU de cada nodo, a su carga de procesos actual y a la conexión de red con los demás nodos, se debe mencionar que bajo el punto de vista de la clasificación de *clusters*, esta tecnología se encuentra en el campo de *clusters* de alto rendimiento, esto debido a que el sistema *Mosix* está diseñado para tomar ventaja del hardware más rápido existente en el *cluster* el cual responde con eficiencia ante cualquier tarea asignada. De esta manera se logra tener una alta capacidad y velocidad de cómputo, pero el proceso interno no es más de un balanceo de carga de tareas en varias máquinas.

Mosix está conformado por algoritmos que monitorizan y responden a las actividades requeridas por el *cluster* mediante la migración automática de procesos hacia los nodos más potentes o más capacitados. Estos algoritmos en *Mosix* son útiles para la distribución automática de procesos, transferencias de procesos desde los nodos más lentos a los más rápidos, balanceo de carga de procesos; entre otros.

La migración de procesos se realiza de manera transparente para el usuario y de forma automática o manual, este proceso consiste en que cuando un nodo se encuentra recargado y con procesos pendientes de ejecución, éstos son migrados a otro nodo disponible, con lo cual se logra que un proceso sea ejecutado en su totalidad en diversos nodos pertenecientes al *cluster*, logrando con ello que no exista ninguna modificación ni pérdidas de información. Es importante hacer notar que cada proceso en el *cluster* tiene un único nodo raíz al cual se encuentra asociado originalmente, cuando se realiza una migración de un proceso hacia otro nodo, éste

se ejecuta como un subproceso del kernel manteniendo siempre la identificación original del nodo raíz al que pertenece originalmente y al cual se le debe retornar el resultado del proceso ejecutado.

2.2.3.3 Beowulf

Es uno de los primeros proyectos de *clustering*, fue construido en el año de 1994 en el Centro de Excelencia en Ciencias del Espacio, Datos e Información *CESDIS* (*Center of Excellence in Space Data and Information Sciences*) bajo la tutela de Thomas Sterling y Don Becker, quienes construyeron un *cluster* de computadoras que interconectaban conjunto de procesadores x86[34] comerciales, los cuales estaban interconectados por una red de tipo Ethernet de 10Mbps, infraestructura a la cual le dieron el nombre de *Beowulf*[35]

Beowulf tuvo mucho éxito luego de su construcción, motivo por le cual fue muy difundido a través de la NASA (*Nacional Aeronautics and Space Administration*), las comunidades académicas y de investigación, con lo cual se convirtió en un sistema muy llamativo y reconocido. Los clusters *Beowulf* dentro de la clasificación del *clustering* pertenecen al grupo de alto rendimiento HPC (*High Performance Computer*)

El sistema *Beowulf* posee una arquitectura la cual consiste en un nodo maestro y uno o más nodos esclavos, los cuales son interconectados por una red ethernet u otra tecnología de interconexión. La principal misión de este sistema es brindar al usuario una única imagen la cual ofrezca eficiencia en la ejecución de una determinada aplicación.

---

[34] x86 es la denominación genérica dada a ciertos procesadores de la familia Intel, a sus compatibles y a la arquitectura básica de estos procesadores, por la terminación de sus nombres: 8086, 80286, 80386 y 80486.

[35] *Beowulf. N*ombre de un héroe de la mitología danesa relatado en el libro La Era de las Fábulas, del autor norteamericano Thomas Bulfinch.

En este sistema el nodo maestro es quien controla al *cluster* en términos generales, éste presta a los nodos esclavos el sistema de archivos para su funcionamiento, a la vez es utilizado como consola de configuración de estos y es quien da la cara al mundo exterior. En la mayoría de los casos los nodos esclavos de un sistema Beowulf son estaciones simples. Los nodos son configurados y controlados por el nodo maestro, y hacen solamente lo que éste le indique.

2.2.3.4 SCYLD

Scyld Beowulf fue desarrollado por la Corporación de Computación *SCYLD*. Es un sistema comercial completo basado en *Beowulf* , está compuesto por un kernel y otras utilidades diseñadas específicamente para hacer *clustering*. Proporciona una imagen única del sistema SSI, esto mediante la herramienta añadida al *kernel* denominada BProc. BProc hace que los procesos ejecutados en los nodos esclavos sean visibles y gestionables desde el nodo maestro. Los procesos arrancan siempre en el nodo maestro y van migrando hacia los nodos esclavos.

2.3 REVISIÓN DE HERRAMIENTAS DE ADMINISTRACIÓN DE CLUSTERS.[19][20][21][22]

2.3.1 GANGLIA

Ganglia es una herramienta *open-source* que en un principio fue desarrollada por la Universidad de Berkley y la división de ciencias computacionales, pero su desarrollo completo se lo hizo en un ambiente universitario donde se logró constituir esta herramienta de uso libre y cuyo funcionamiento no dependía de componentes pertenecientes a otros propietarios. Todos los datos en Ganglia se transmiten utilizando XML y XDR mediante una conexión TCP y *multicast.*

Ganglia es un software que permite realizar un monitoreo en tiempo real para clusters, y ha sido utilizado en ambientes universitarios, gubernamentales e implementaciones en áreas comerciales. El monitoreo realizado con Ganglia es el mismo cuando se trata de pocos nodos o cuando se tiene un número mayor de estos.

Desde el momento de la creación de esta herramienta de monitoreo, se han desarrollado varias versiones, las cuales se puede citar a Ganglia 1.0, quien apareció en el año 2000, basa su funcionamiento en demonios escritos en *Perl*; también se puede mencionar a Glanglia Versión 2.x, quien aparece en el año 2001, la cual es una versión más estable y escrita en lenguaje C; y a partir del año 2002 el desarrollo de Ganglia estuvo a cargo de *SourceForge*[36] que desarrollaron una versión de Ganglia multiplataforma, que se ejecuta sobre varios sistemas operativos dentro de los cuales de puede citar a Linux, FreeBSD, Solaris, Windows; entre otros.

En Enero del 2002 aparece Ganglia versión 2.5, la cual fue desarrollada para ambientes cliente-servidor y su funcionamiento principal se basa en la existencia de dos demonios y una interfaz Web. Los demonios son: el *gmond* (*Ganglia Monitoring Daemon*) y *gmetad* (*Ganglia Meta Daemon*) y la interfaz Web es el *Ganglia Web Frontend*.

- *gmond*. Demonio multi-hilo[37] utilizado para monitoreo, el cual se ejecuta en cada uno de los nodos del cluster que se quiere monitorear.

- *gmetad*. Demonio utilizado para la recolección de datos, es instalado en el nodo de administración, su función es la de recolectar la información vía XML que los nodos monitoreados le envían en intervalos regulares de tiempo, *gmetad* recoge toda la información y la guarda en una base de datos *Round-Robin*.

  Una base datos *Round-Robin* no es mas que una base de datos circular, la cual va a contener siempre la misma cantidad de datos, debido a que tiene almacenada toda la extensión de la base de datos, simplemente sobrescribe los datos antiguos. Una manera mas sencilla de entender este tipo de base de datos es tener un circulo en el que se van a ir colocando datos. Si se empieza por colocarlos en un determinado punto, cuando se haya dado una vuelta a

[36] Visitar sitio web: http://*www.sourceforge.net*

[37] Multi-hilo.se refiere a que dos o más tareas se ejecutan "aparentemente" a la vez, dentro de un mismo programa.

todo el círculo, se llega al inicio del mismo, y es ahí cuando se empieza sobrescribir los datos recopilados al principio. Este sistema permite almacenar y representar datos en intervalos temporales.

Esta base de datos que no crece en el tiempo y permite crear gráficas para representar la información almacenada. Es importante mencionar que el demonio *gmetad* se encarga de concatenar los datos XML proporcionados por el cliente, esto con la finalidad de compartir esta información con el servidor Web u otro *frontend* que este ejecutando el demonio *gmetad.*

- *ganglia frontend.* Es el demonio encargado de mostrar la información en una página Web dinámica en tiempo real, la cual actualiza su información durante un intervalo de tiempo determinado. Ganglia se muestra como una interfaz html, pero se debe hacer notar que presenta información de tipo XML, este demonio presenta datos (CPU, memoria, velocidad de CPU's; entre otros) de los nodos del *cluster* en tiempo real y de manera gráfica.

Es importante mencionar que ganglia presenta otras herramientas importantes para realizar un adecuado monitoreo, dentro de estas se puede citar a *gmetric*, demonio que permite monitorear a un determinado equipo de acuerdo a métricas establecidas, ya sean estas número de procesadores, memoria de CPU; entre otras. Otro demonio que es impotente es el *croad*, el cual permite especificar la realización de una determinada tarea a cierta hora y día que se determine.

2.3.2 WEBMIN

Webmin es un sistema utilizado para la configuración, administración y monitorización remota de sistemas y servidores. Es una herramienta web que presenta la información en formato HTML en cualquier navegador que se este utilizando, esto es posible debido a que el propio *webmin* posee un servidor web pequeño el cual presenta al administrador un menú con una serie de configuradores y monitores para los diferentes programas y servicios que se estén ofreciendo en el sistema. Estos

configuradores son representados en forma de un *script CGI (Common Gateway Interface)* el cual es un método para permitir la interacción de programas ejecutables entre el cliente y el servidor. Un script CGI es ejecutado en tiempo real, lo que permite manipular la información de forma dinámica. Por ejemplo, se quiere conectar bases de datos al World Wide Web para permitir que las personas de cualquier parte la manipulen.

La instalación de esta herramienta de monitoreo y configuración es sencilla, sus configuradores y su servidor Web están escritos en *Perl*, siendo el único requisito para su funcionamiento la utilidad de sus paquetes y librerías. Webmin es un software de mucha utilidad, permite monitorear servicios tales como el Web, SMTP, DNS; entre otros, los cuales se monitorean por defecto al momento de instalar Webmin, pero esta herramienta permite añadir nuevos configuradores para que sean monitoreados y configurados, como es el caso de un cluster LVS.

Luego de la instalación completa de esta herramienta, se pueden administrar y monitorear las partes más comunes del sistema operativo, tales como: Crear - borrar usuarios y grupos de ellos, ver qué procesos se están ejecutando, programar trabajos a un tiempo determinado, asignar cuotas de disco, reiniciar o apagar el equipo; entre otros. También se pude tener el control de servidores de DNS, Apache, SSH, Squid Proxy, Samba, MySQL; entre otros. La configuración de servicios de red es algo que también se puede controlar bajo esta herramienta, la administración de los servicios de red, la configuración de las interfaces, las direcciones IPs; son algunas acciones que se pueden realizar. Webmin también permite administrar el hardware del servidor, trae una opción para monitoreo y configuración de un *cluster*, para el cual se debe instalar los configuradores respectivos.

2.3.3 LVS MANAGER

El software Linux Virtual Server Manager (LVSM) es una herramienta de configuración y administración de un *cluster* LVS por medio de una interfaz HTML.

LVSM esta programado en *Perl* y necesita del módulo *mod_perl*[38] de Apache[39] para operar correctamente.

LVSM para su funcionamiento se basa de dos componentes: el uno es *plvsd*, el cual es un demonio que se instalará en cada uno de los servidores del *cluster* y se encarga de mantener el estado y configuración de cada equipo del *cluster*, así como presenta algunas herramientas para la alta disponibilidad tales como *heartbeat, mon*; así como herramientas para configuración de servidores reales.

El segundo componente es el *LVSM Web GUI*, interfaz web mediante la cual se puede configurar el *cluster*. Presentará una serie de formularios HTML con las opciones de configuración, estas serán procesadas por unos *scripts CGIs* desarrollados en Perl, los cuales se comunicarán con el demonio plvsd de cada servidor para acceder a modificar la configuración de cada máquina.

Una vez instalado *LVSM* se puede realizar las siguientes acciones: Balanceo de carga, alta disponibilidad, monitoreo y administración de servicios a nivel local y remota de servidores web, estadísticas de funcionamiento del *cluster*; entre otras.

2.3.4 C3

La herramienta de administración C3 (*Cluster Command & Control*) está basada en la utilización de un conjunto o líneas de comandos los cuales son utilizados para ejecutar tareas de administración en los servidores del cluster.

C3 fue desarrollado por el Laboratorio Nacional Oak Ridge y es una herramienta de distribución libre, permite realizar diferentes acciones como las detalladas a continuación:

- Ejecutar comandos del sistema en forma distribuida.

---

[38] *Mod_perl. E*s un software instalado en Apache, o sea el servidor de páginas web, que permite introducir en los ficheros .html código desarrollado en Perl.

[39] servidor http

- Distribuir y buscar archivos.
- Eliminar procesos que estén corriendo.
- Reiniciar y apagar el cluster en forma remota.
- Generar imágenes para actualizar el cluster.

2.4 HERRAMIENTAS PARA PLANIFICACIÓN DE TAREAS.[23][24][25]

Existen varias herramientas de planificación de tareas disponibles en la Web, pero las más utilizadas y conocidas son las analizadas a continuación:

2.4.1 PBS

PBS (*Portable Batch System*) es una herramienta de planificación de tareas desarbolada en el año de 1999 por la empresa Veridian (www.veridan.com) quien realizó el sistema PBS para la NASA y con ello administrar sus recursos computacionales.

PBS permite monitorear y controlar trabajos por lotes en uno o varios sistemas. El trabajo por lotes hace referencia a que un trabajo será calendarizado con la finalidad de ejecutarse en un tiempo determinado basado en ciertas políticas y disponibilidad de recursos.

Es importante mencionar a que se hace referencia con trabajo, éste típicamente es un *shell*[40] *script*[41] y un conjunto de atributos que proveen información de recursos y control acerca del trabajo.

PBS contiene tres procesos demonio que conforman su estructura:

---

[40] *shell* - es el intérprete de comandos de UNIX el cual recoge todo lo que se ingresa por teclado y lo convierte en un programa que se ejecuta.

[41] Un *shell script* es un archivo de texto, que contiene órdenes de shell, la cual va ejecutando una a una siguiendo un guión preestablecido en el propio script.

- Servidor de trabajos (*pbs_server)*. Servidor que cumple la principal función del sistema PBS, ya que su misión es brindar los servicios básicos para recibir, crear, ejecutar y modificar trabajos *batch*[42].
- Planificador (*pbs_sched*). Demonio encargado de manejar las políticas de ejecución de los trabajos, éste se encarga de decidir donde y cuando colocar un trabajo en ejecución, las políticas de ejecución deben ser preestablecidas por el administrador del sistema mediante el servidor de trabajos.
- Ejecutor de Trabajos (*pbs_mon*). Demonio que se encarga de la ejecución de los trabajos, éste recibe el trabajo a realizarse, pero previamente crea una copia con una sesión del usuario solicitante. Una vez realizado el trabajo encomendado, su resultado lo envía al servidor de salida.

PBS es una herramienta que ofrece muchas ventajas para la planificación de tareas, entre las cuales se pueden indicar que, es soportado en sistemas operativos Windows 2000 y XP, así como de UNIX y Linux, provee de múltiples interfaces gráficas de usuario que son útiles para realizar las peticiones por lotes y su respectiva planificación, proporciona mecanismos de seguridad y control de acceso al sistema, presenta un informe detallado de los eventos realizados en el mismo, provee una herramienta de monitoreo gráfico del sistema distribuido, permite priorizar la ejecución de tareas dependiendo de las necesidades de los usuarios; entre otras.

2.4.2 CÓNDOR

Condor es una herramienta de código abierto utilizada para la planificación de tareas, fue desarrollada por el proyecto de investigación CONDOR de la Universidad de Wisconsin Madison y el Departamento de Ciencias de la Computación. Esta herramienta posee soporte para sistemas Linux, Windows, UNIX; entre otras.

Condor es un especializado sistema de planificación, administración y monitorización de trabajos que se están ejecutando, además provee un administrador adecuado

[42] *trabajo batch* – es un guión del shell con el conjunto de comandos que se quieren ejecutar en el sistema, además esta compuesto de varias directivas que le indican a PBS las características (atributos) del trabajo, y recursos solicitados (tal como memoria o tiempo de CPU).

para manejo de colas, mecanismos de planificación y prioridad , monitoreo y administración de recursos.

Condor recibe los trabajos requeridos por los usuarios y los ubica en una cola, éste mediante políticas preestablecidas determina donde y cuando se ejecutarán, en este proceso CONDOR presenta un monitoreo de dicha ejecución e informa al usuario final cuando termino el trabajo que se estaba realizando.

Condor presenta muchas características las cuales lo hacen muy útil, las más importantes y relevantes se resumen a continuación:

- Prioridad de Trabajos. Los usuarios pueden asignar prioridades para sus trabajos, y con ello poder controlar el orden de ejecución de los mismos.

- Prioridades de usuario. Los usuarios pueden tener prioridades de ejecución, previa asignación de los administradores del sistema basados en mecanismos de repartición justa y un orden adecuado de ejecución.

- Dependencia de trabajos. Algunos conjuntos de trabajo requieren un orden de dependencia entre trabajos, es decir unos trabajos pueden empezar su ejecución solo si otros ya fueron ejecutados de una manera satisfactoria y siguiendo un orden adecuado.

- Migración y puntos de control de trabajos (*checkpoints)*. Condor puede de manera transparente realizar un punto de control o *checkpoint* del trabajo que se este ejecutando y con ello obtener un detallado informe de su estado. Una vez realizado el punto de control, el trabajo puede continuar ejecutándose desde el momento donde este se realizo. Además esta característica permite realizar el migrado de trabajo de una máquina a otra.

- Puntos de control periódicos. Condor tiene la capacidad de configurar puntos de control periódicos para un determinado trabajo que se este ejecutando. Esta característica es de mucha utilidad para salvaguardar el tiempo

acumulado de computación en un trabajo y con ello reducir las perdidas que ocasionan las fallas de hardware y software.

- Resumen y suspensión de trabajos. Condor tiene la ventaja de preguntar al sistema operativo si puede suspender un determinado trabajo y luego extraer un resumen detallado del estado del mismo.

El funcionamiento de CONDOR depende de la existencia de varios demonios los cuales cumplen diversas funciones que se detallan a continuación:

- Condor_master. Este demonio tiene la principal función de simplificar la administración del sistema. Es responsable de garantizar el funcionamiento del resto de demonios que se están ejecutando en el resto de máquinas existentes en el sistema. Condor_master está constantemente chequeando nuevas fuentes binarias para que sean actualizadas a los demonios que está manejando, si estas actualizaciones se encontraron para un determinado demonio, el master luego de realizar la actualización binaria debe reiniciarlo. Esta ventaja permite a Condor estar actualizado fácilmente y constantemente, e incluso cuando ocurrió algún inconveniente presentado con algún demonio, Condor_master envía un e-mail al administrador indicando lo sucedido y el inconveniente presentado. Otra ventaja que presenta este demonio está en que automáticamente puede reconfigurar, parar, iniciar los demonios que se están ejecutado remotamente.

- Condor_startd. Este demonio es instalado en cada máquina perteneciente al conjunto de máquinas Condor. Este se encarga de advertir a la directiva *ClassAd* (Esta directiva proporciona un marco de trabajo el cual permite determinar si algunas solicitudes de acceso a recursos (trabajos) coinciden con los recursos ofrecidos por el sistema(computadores)) que contiene atributos relacionados a las máquinas que están trabajando. Ejecutándose el demonio *startd* habilita a una máquina para ejecutar un determinado trabajo en base a los atributos proporcionados y el responsable de hacer cumplir la política bajo la cual los trabajos serán iniciados, suspendidos o reiniciados.

- Condor_starter. Este demonio se encarga de preparar el ambiente de ejecución y supervisar el trabajo una vez que esté funcionando. El demonio *starter* detecta la terminación del trabajo que se está ejecutando y envía información del estado de la máquina donde se ejecutó.

- Condor_schedd. Demonio encargado de hacer cumplir el trabajo en las máquinas de Condor, cualquiera de ellas permite a los usuarios someter trabajos para que sean ejecutados, pero para ello deben tener en sus máquinas ejecutándose el demonio *schedd*, una vez sometidos los trabajos son cargados en una cola de trabajos para que sean atendidos. Existen algunas herramientas dentro del paquete Condor que permiten conectarse con el *Condor_schedd* las cuales permiten manipular las colas de trabajo.

- Condor_shadow. Demonio encargado de hacer la llamada al sistema de la máquina a la cual se está sometiendo al trabajo y su resultado es enviado vía la red del sistema.

- Condor_collector. Este demonio es el encargado de recolectar toda la información relacionada al estado de las máquinas administradas por Condor. Todos los demonios periódicamente envían información relacionada a su estado.

2.4.3 MAUI

MAUI es una herramienta netamente utiliza como planificador de tareas, muy utilizada en ambientes de clustering, fue desarrollada por el Centro de Computación de Alto Desempeño, el Laboratorio Nacional del Noreste del Pacífico, el Centro de Supercomputación de San Diego y el Laboratorio Nacional Argonne.

Esta herramienta de planificación es muy utilizada para establecer un gran número de políticas de planificación (scheduling), con las cuales se puede lograr obtener un a mayor eficiencia y desempeño de los sistemas. MAUI es una herramienta de libre distribución y su código fuente se encuentra disponible. Esta herramienta no puede

ser integrada como planificador en cualquiera de los sistemas de colas como Condor, PBS; entre otras, con las que tiene una total incompatibilidad.

MAUI posee muchas ventajas para la planificación de tareas, dentro de las mas importantes se puede mencionar la posibilidad de implementar calidad de servicio QoS a ciertas tareas, definición de políticas, planificación, prioridades y configuración de tareas, informes detallados de utilización de recursos; entre otras.

Bibliografía Capítulo 2

1. "*Linux System Administration I: Implementation*", Guía del estudiante. ACE Advanced Career de IBM. Unidad 10.

2. http://www.wikilearning.com/administracion_de_archivos_ii-wkccp-9574-29.htm, "Administración de archivos".

3. http://www.cecalc.ula.ve/HPCLC/slides/day_07/practica_pvfs.pdf#search=%22pvfs%22, *"Parallel Virtual File System (PVFS)".*

4. http://www-128.ibm.com/developerworks/aix/library/au-unixreiserFS/index.html, "*ReiserFS".*

5. http://samba.anu.edu.au/rsync/, "*rsync",* ultimo *release* 6 de Noviembre del 2006.

6. http://www.namesys.com/v4/v4.html

7. www.info-ab.uclm.es/asignaturas/42577/docs/presTema5.pdf#search =%22diferencias%20 entre%20Andrew%20file%20system%20y%20coda%22, "Sistemas de archivos *CODA".*

8. http://www.ntfs.com/, "*NTFS File System"*, 1997 – 2007.

9. http://www.coda.cs.cmu.edu/, "*CODA File System".*

10. http://www.ant.org.ar/cursos/curso_intro/filesystem.html, "Estructura del sistema de archivos Linux".

11. http://unthought.net/Software-RAID.HOWTO/Software-RAID.HOWTO.spanish-1.html#ss1.4, "*RAID´s"*

12. http://www.smdata.com/NivelesRAID.htm, “Niveles de RAID”, última actualización 20 de Marzo del 2007.

13. http://xcr.cenit.latech.edu/ha-oscar/papers.html., “*HA-OSCAR”.*

14. http://www.linuxvirtualserver.org/Documents.html, *“LVS-Documentation”,* última actualización 12 de Marzo del 2007.

15. http://www.linuxvirtualserver.org/HighAvailability.html, *“High Availability”,* última actualización 12 de Marzo del 2007.

16. http://www.supersparrow.org/, “*SuperSparrow”,* última actiualización 13 de Noviembre del 2006.

17. http://www.mosix.org/, *“MOSIX Cluster and Grid Management”*.

18. http://www.beowulf.org/, *“Cluster Beowulf”.*

19. http://ganglia.sourceforge.net/, *“Ganglia Documentation”.*

20. http://www.webmin.com/, “*Web-based System Administration”.*

21. http://www.csm.ornl.gov/torc/C3/. “*Project C3 – Cluster Command and Control”,* última modificación 5 de enero de 2007.

22. http://www.csm.ornl.gov/torc/C3/Papers/C3_DAPSYS2000.pdf, “*Cluster Command & Control (C3) Tools Suite”,* 3rd Distributed and Parallel Systems (DAPSYS 2000), September 10-13, 2000, Balatonfüred, Lake Balaton, Hungary. Published by: Kluwer Academic Publishers, ISBN: 0-7923-7892-X

23. http://www.openpbs.org/, "*Portable Batch System"*, 2003 Altair Grid Technologies.

24. http://www.cs.wisc.edu/condor/, "*CONDOR High Throughput Computing".*

25. http://www.usenix.org/publications/library/proceedings/als00/2000papers/papers/full_papers/bode/bode_html/, *"The Portable Batch Scheduler and the Maui Scheduler on Linux Clusters"*, última modificación 8 de septiembre de 2000.

CAPÍTULO 3
DISEÑO, INSTALACIÓN Y CONFIGURACIÓN DEL *CLUSTER*

3.1 OTRAS IMPLEMENTACIONES REALIZADAS.

Dentro del grupo de implementaciones realizadas en el mercado por fabricantes que hacen uso de herramientas libres, existen muy pocas publicaciones, y con ello su información no muy difundida, pero a continuación se presenta un resumen de las principales características:

3.1.1 CLUSTER GOOGLE [1]

Esta implementación es una gran muestra de la necesidad de un sistema de balanceo de carga y alta disponibilidad. Google atiende a más de 5,000 millones de búsquedas cada mes[43], y esta cifra aumenta progresivamente cada día. Para atender todas estas peticiones, y buscar entre más de 3,000 millones de documentos, Google optó por la tecnología Linux.

Google posee un *cluster* de cerca de 20,000 servidores ubicados y repartidos en siete centros de datos en los diferentes puntos del planeta, estos centros se encuentran en Washington D.C. (USA), Herndon (Virginia, USA), Santa Clara (California, USA) o Zurich (Suiza).

Entre los centros de datos, Google utiliza su propio gestor de tráfico y su propio software de balanceo de carga, para dirigir cada petición hacia el mejor servidor, el balanceo de carga lo realiza en base a DNS entre varios *clusters,* las peticiones se resuelven de manera independiente y de forma paralela, existe redundancia en diferentes niveles ya sean estos a nivel de alimentación eléctrica, arreglo de discos en los servidores (RAID), alta disponibilidad mediante la replicación masiva

---

[43] Ver sitio Web: *http://google.dirson.com/tecnologia.php*

de los servicios críticos y la utilización de múltiples servidores con hardware de bajo costo.

Cada PC posee uno o dos discos duros de 40Gb ó 75Gb, de marca IBM, además utiliza la técnica de almacenamiento de datos distribuido, ésto con la finalidad de evitar tener puntos de falla únicos.

Cada máquina utiliza el sistema operativo Linux RedHat, esto gracias al convenio realizado en mayo de 2002 en el que Google llegó a un acuerdo con *Red Hat* para que esta empresa le proporcionase el software del Sistema Operativo.

3.1.2 VA-LINUX [2]

Proyecto que forma parte de *VAnessa*[44] (*VA Network Enhanced Scalable Server Architecture*), es una solución basada en *Ultra monkey*[45], caracterizada por usar software libre como implementación y la empresa se dedica a dar servicio sobre ese software, no incluye software comercial, este proyecto permite garantizar los siguientes servicios:

- Uso de LVS para balanceo de carga.
- Fácilmente escalable a un gran número de servicios que dependan de IP.
- Monitorización de los servicios con Ldirectord. [46]
- Usa Heartbeat[47] para proveer alta disponibilidad.

3.1.3 LIFEKEEPER [3]

Herramienta creada para solventar los problemas de falta de disponibilidad en empresas de negocios, ya que este inconveniente provoca muchas pérdidas para éstas, es por ello que *Lifekeeper* desarrollo un software llamado SteelEye's para Linux, el cual es una aplicación que provee una alta disponibilidad a los sistemas

---

44 Ver la página : *http://www.vergenet.net/linux/vanessa/*

45 Revisar Capítulo 2 sección 2.2.2.6.3.

46 Revisar Capítulo 2 sección 2.2.2.6.

47 Revisar Capítulo 2 sección 2.2.2.6.

para que se encuentren operativos la mayor parte del tiempo y ésto lo logra mediante un monitoreo adecuado de la conectividad existente en el sistema.

LifeKeeper evita tener puntos únicos de falla ya que permite tener una configuración que permite de manera automática recuperar el sistema cuando este cae, esto debido a que los servidores del *cluster* se encuentran monitorizados y los unos son respaldos de los otros, todo gracias a la ventaja que brinda el demonio *Heartbeat* el cual provee una alta disponibilidad al sistema pudiendo con este configurar los nodos directores en estado activo-activo o activo-pasivo. Con esto es importante tomar en cuenta la recuperación de datos que en este proyecto permite realizarlo de tres maneras:

- Recuperación en multi – direcciones. Estructura donde se tiene en un anillo de alta velocidad tanto a los servidores principales como los de respaldo.

- *Failovers* en cascada. Se tiene varios servidores en cascada, uno es el respaldo del siguiente y así sucesivamente, todos ellos interconectados por una red de alta velocidad.

- Soporte compartido de datos. Se tiene un servidor de respaldo de todos los existentes, el problema de esta configuración radica en la concentración de funciones en el equipo de respaldo, con lo cual se puede tener un punto de fallo.

Una de sus principales ventajas está en que mientras se realiza cualquier actualización o trabajos en los servidores, se lo puede realizar sin necesidad de parar los servicios que se están ejecutando, logrando con ello tener una disponibilidad aún mayor, es importante mencionar la escalabilidad que brinda esta herramienta, ya que permite añadir nodos al *cluster* dependiendo de las necesidades del usuario y el sistema.

3.1.4 MISSION CRITICAL LINUX [4]

Creada por la compañía *Mission Critical Linux* de manera gratuita, la cual posee mejoras en el *kernel* de Linux para proveer un ambiente de alta disponibilidad en los servicios que determinada empresa este ofreciendo.

La herramienta que va permitir alta disponibilidad es la tecnología *kimberlite*[48], la cual ha realizado mejoras al *kernel* de Linux con el que provee soporte para dos nodos servidores conectados a un disco compartido SCSI o a fibra óptica. Si uno de los dos nodos deja el cluster el otro lo detecta y arranca unos scripts de recuperación que hacen tareas necesarias para reiniciar las aplicaciones en el nodo que queda, el monitoreo se realiza en base a demonios *heartbeat* los cuales detectan la disponibilidad de los servidores.

Una vez recuperado el nodo principal se vuelve a unir al cluster, los programas pueden ser migrados otra vez a él, manualmente o automáticamente, si se requiere. Está especialmente diseñado para mantener la integridad de los datos y ser muy robusto. El inconveniente de esta técnica de alta disponibilidad radica en el precio del hardware pues una interfaz externa SCSI o sobre fibra óptica es alto costo, además el problema de desaprovechar uno de los dos servidores de respaldo cuando se trabaja con la configuración activo – pasivo conlleva a tener una subutilización del mismo.

3.2 ANÁLISIS DE REQUERIMIENTOS PARA CONFIGURACIÓN DE CLUSTERS.

3.2.1 REQUERIMIENTOS DE HARDWARE.

Los requerimientos de hardware para el *cluster* de balanceo de carga y alta disponibilidad en términos generales se detallan a continuación:

---

48 Ver sitio Web: *http://www.missioncriticallinux.com/projects/kimberlite/kimberlite.pdf*

- Un cliente el cual debe acceder desde el Internet o Intranet.
- Un *router* el cual va a ser el *gateway* del *cluster*.
- Un servidor director y un *standbye* con tarjeta de red, sobre la cual se va configurar la dirección IP virtual, la dirección IP Real.
- Un *switch* de comunicaciones, el cual va a unir los servidores reales con el nodo director.
- Varios servidores reales los cuales van a tener una tarjeta de red en las que se configuraran direcciones IP reales mediante las cuales se conectan con el servidor director.

Es importante tomar en cuenta las necesidades mínimas de hardware para configurar *HA-OSCAR* y *LVS*, así como del costo de los componentes que estos requieren, a continuación se lista los requerimientos mínimos necesarios para estos paquetes de configuración:

- Requerimientos mínimos para *HA-OSCAR.* Uno de sus principales requerimientos es que todos los computadores a utilizarse deben ser homogéneos, es decir que debe tener el hardware idéntico, en especial sus tarjetas de red (*ethernet 10/10/1000*), igual número y capacidad de procesadores, la utilización de *SCSI* o *IDE* para los sistemas de archivos principales.

  Para los servidores principal y *standbye* se requiere mínimo procesadores pertenecientes a la arquitectura x86, la capacidad en disco duro se detalla en la Tabla 3-1:

Almacenamiento	4 GBytes
Sistema operativo Linux, HA-OSCAR y todas las aplicaciones que vienen incorporadas en este paquete	6 GBytes
Imagen Sistema operativo Linux	6 GBytes

Tabla 3-1: Capacidad Disco Duro para Servidor Director

Es importante mencionar que la cantidad necesaria de almacenamiento que el cluster necesita depende también del tamaño de las aplicaciones a ejecutarse. El tamaño de memoria RAM necesaria es de 512 MBytes.

- Requerimientos mínimos para *LVS.* Para el nodo director, se requiere mínimo un computador *Pentium I* de 75 MHz con una tarjeta de red *ethernet* de 100 Mbps, a nivel de requerimientos de disco, es suficiente tener los 4 GBytes para la instalación del sistemas operativo Linux, dentro del cual utilizando la versión del *kernel* 2.6 viene ya incluido el módulo IPVS para el balanceo de carga y todas las aplicaciones que éste hace uso para su funcionamiento. Es importante también dimensionar la capacidad de disco requerida dependiendo del tipo de aplicaciones a ejecutarse. El tamaño de memoria RAM necesaria es de 512 MBytes.

  Los servidores reales, también poseen los mismos requerimientos, tomando en cuenta los requerimientos necesarios para la instalación y funcionamiento de las aplicaciones que el *cluster* va a ofrecer.

Tomando en cuenta estas consideraciones, a continuación se lista las características de hardware que cumplen los requerimientos mínimos de configuración con los cuales se esta desarrollando el *cluster* de balanceo de carga y alta disponibilidad.

3.2.1.1 Información general - servidores

NOMBRE DEL SISTEMA	SERVIDOR DIRECTOR
PROCESADORES	INTEL (R) PENTIUM (R) 4 CPU 2.8 GHZ
MEMORIA EN MB	512 MB DDR
TARJETA DE RED	FAST ETHERNET PCI CNET PRO 10/100 Mbps

Tabla 3-2. Información General de Servidores

3.2.1.2 Discos duros

MODELO	CAPACIDAD
SAMSUNG-IDE\DISKSAMSUNG_SP1203N	120 GBytes
SATA	120 GBytes

Tabla 3-3: Información Discos Duros

3.2.1.3 Tabla de particiones

PARTICION	TAMAÑO EN MB	TYPO DE PARTICION	PUNTO DE MONTAJE
/dev/hda1	100	ext3	/boot
/dev/hda2	19100	ext3	/
/dev/hda3	80	ext3	/dev/shm

Tabla 3-4: Tabla de particiones

3.2.1.4 Información de dispositivos de red

DISPOSITIVO	MARCA	MODELO	DESCRIPCION
Switch	3COM	3C1670108	Switch de 8 puerto 10/100 Mbps con un puerto uplink Gbps

Tabla 3-5: Información de dispositivos de red

3.2.2 REQUERIMIENTOS DE SOFTWARE.

Los requerimientos de software se detallan a continuación:

- Sistema Operativo Linux. A nivel de nodos directores como servidores reales pueden utilizarse las diferentes versiones de software libre como Red Hat 9.0, Mandrake10.0, CentOS 4, *Fedora Core* 3.0 o 5.0; entre otras existentes. Finalmente se utilizó el sistema operativo *Fedora Core* 5.0.

- Nodos Directores. Se requiere la herramienta *HA OSCAR* 1.2 para la alta disponibilidad, IPVS para el balanceo de carga, herramienta que viene ya incluida en el *kernel* versión 2.6 del sistema operativo *Fedora Core* 5, se necesita el paquete de configuración *ipvsadm*, herramienta que permite la administración y configuración de LVS, además se necesita de la herramienta *webmin* con la cual se puede configurar y monitorear *HA OSCAR.*

- Servidores Reales. Se requiere instalar *Postfix* con el cual se va proporcionar el servicio de correo electrónico, *SUN Java Studio Creator*[49] para el desarrollo de la aplicación Web, *BIND* para la configuración del servicio de resolución de nombres DNS.

- Clientes. En los clientes se requiere cualquier sistema operativo, ya sea Windows o Linux, además de requiere instalar *MySQL Server y Java EE SDK* ya que el generador de tráfico *Ecperf* requiere de una base de datos y un compilador java para su funcionamiento.

3.3 DISEÑO Y CONFIGURACIÓN DEL *CLUSTER*.

Previo al diseño e instalación del *cluster* de balanceo de carga y alta disponibilidad se debe dar un nombre al sistema, para lo cual se lo denominó de “ *Cluster EPN”.* Posteriormente se debe hacer un pequeño análisis de los mecanismos y alternativas de configuración del *cluster* que Linux Virtual Server

[49] Ver sitio Web: http://www.sun.com/software/

LVS y OSCAR HA poseen para términos de balanceo de carga y alta disponibilidad respectivamente.

3.3.1 LVS – CONSIDERACIONES PREVIAS A LA CONFIGURACIÓN E INSTALACIÓN.

Primeramente se va realizar una revisión general de los aspectos generales que involucra el utilizar LVS. LVS es un grupo de servidores reales y un servidor director el cual es el encargado de dar la cara al mundo exterior. LVS puede ofrecer muchos servicios ya sean estos de alta capacidad, redundantes, los cuales están disponibles desde cualquier servidor simple. Un servicio esta definido como una conexión para un puerto simple, por ejemplo telnet, http, https, ssh; entre otros.

La semántica cliente-servidor se mantiene, cada cliente cree que se conecta directamente con el servidor real, así como cada servidor real piensa que se conecta directamente con el cliente. Ningún cliente y servidor real sabe que en la conexión participo un servidor director.

3.3.1.1 Determinación del mecanismo de Balanceo de Carga.

Como ya se analizó en la sección y capítulos anteriores los mecanismos de balanceo de carga son tres, por lo cual se debe hacer un análisis de las ventajas y desventajas de cada uno de ellos con la finalidad de poder determinar cual de ellos se acopla para las necesidades del presente proyecto de titulación.

3.3.1.1.1 Ventajas y desventajas de LVS – NAT

Ventajas.

- Los servidores reales poseen una sola tarjeta de red configurada, mediante la cual se conectan entre si y con el servidor director por medio del cual

dirigen la respuesta del trabajo encomendado por el cliente. Esta es una ventaja debido a que se omitiría el costo de instalar otra tarjeta de red como ocurre en los otros métodos de balanceo de carga.

- Es muy útil para cuando se tiene un sistema por el cual no circula gran cantidad de tráfico, es decir con aplicaciones que no tengan gran demanda de acceso.

Desventajas.

- La principal desventaja de este método radica en el problema que genera la sobrecarga que tendría el nodo director al tener que reescribir todos los paquetes de datos originados desde y hacia los servidores reales y clientes, problema que se hace notorio cuando el número de peticiones y respuestas es muy grande, razón por la cual hace necesario invertir un mayor costo en la obtención de un equipo más potente para con ello lograr que el rendimiento de este no se vea afectado.

- El nodo director no permite temer muchas conexiones con los servidores reales, esto debido al inconveniente ya mencionado relacionado a la potencia de cálculo del director y al ancho de banda necesario para las conexiones con éste.

- Otra desventaja radica en el principal punto de falla que se convierte el nodo director, debido a que por este cruzan tanto las peticiones y respuestas originadas por y para clientes.

3.3.1.1.2 Ventajas y desventajas de LVS – Encapsulamiento por IP.

Ventajas.

- El balanceo de carga basado en encapsulamiento IP presenta una mayor escalabilidad que la que presenta el balanceo realizado por NAT. Éste

permitirá escalar hasta un mayor número de servidores, 100 o más, con la condición de que todos soporten encapsulado IP (IP *tunneling*). Este incremento de escalabilidad se da ya que el nodo director únicamente participa el momento en que el cliente solicita algún requerimiento, realiza el respectivo encapsulamiento IP y lo reenvía al servidor real disponible, este se encarga de procesarlo y enviar el resultado directamente al cliente sin necesidad de hacerlo mediante el nodo director, logrando con ello evitar saturarlo e impedir que exista un cuello de botella sobre en este.

- Este mecanismo de balanceo de carga posibilita distribuir los servidores reales a lo largo de una red de área amplia en lugar de tenerlos todos en un mismo segmento de red local. Esta ventaja es de mucha importancia para cuando no se quiere concentrar toda una infraestructura en un mismo sitio, sino tenerla distribuida en diferentes lugares, logrando con ello tener redundancia del sistema.

Desventajas.

- Una de las desventajas radica en que los dispositivos utilizados en este tipo de balanceo requieren que todos los dispositivos soporten el protocolo IP *tunneling* o IPSec.

- Un inconveniente que presenta este método de balanceo de carga cuando los servidores reales y director se encuentran en una misma red, se da debido a que poseen la misma dirección IP virtual, va existir problemas de conflictos de dirección IP, esto debido a los servidores reales y el director responderían a requerimientos ARP, logrando con ello tener un conflicto interno, haciendo que los paquetes sean enviados tanto al director como a los servidores reales y ese no es el propósito, la idea es de que al nodo director o balanceador le lleguen todas las peticiones y sea este quien debe responder a los requerimientos ARP y re - direccionarlas a los servidores reales. Para ello se debe lograr que los servidores reales que se encuentran en la misma red del nodo director no deban responder a

requerimientos ARP y procesar los paquetes destinados a la IP pública de manera local, ello se logra ocultando las interfaces donde se tiene configurada mencionada dirección real.

- Se podría considerar una pequeña desventaja debido a que todos los servidores deben poseer dos tarjetas de red configuradas, haciendo de esta manera que los costos de montaje a nivel de hardware aumenten.

3.3.1.1.3 Ventajas y desventajas de LVS – Enrutamiento directo.

Ventajas.

- Una de las ventajas de esta técnica radica en que el nodo director no va sobrecargar su trabajo debido a que el enrutamiento se realiza a nivel MAC, evitando con ello realizar trabajos adicionales como encapsulamiento IP y traslaciones NAT, las cuales consumen recursos del servidor haciendo que este llegue al máximo de su capacidad y se sature.

Desventajas

- Mecanismo afectado por el mismo problema que presenta el balanceo por enrutamiento IP, debido a que los servidores reales y el director poseen la misma dirección IP virtual o pública, teniendo con ello problemas de conflictos internos. Para solventar este problema se debe lograr que solo el nodo director responda a los requerimientos ARP mientras que los servidores reales deben tener sus interfaces ocultas para no responder a estos requerimientos.

- Por el hecho de que el nodo director realiza el re – direccionamiento de las peticiones a nivel MAC, imposibilita el tener el *cluster* disperso en áreas geográficas extensas ya que este método requiere que tanto nodo director como servidores reales estén en el mismo segmento físico.

Una vez realizado el respectivo análisis de cada uno de los mecanismos de balanceo de carga y de acuerdo a las necesidades que la aplicación a implementarse requiere, se decide utilizar el método de balanceo basado en enrutamiento directo debido a las siguientes razones:

- Por el hecho de tener toda la infraestructura en un mismo sitio se descartaría la posibilidad de hacerlo mediante el mecanismo de encapsulamiento IP, así como evitar los problemas de sobrecarga de procesamiento en el servidor director por concepto de enmascaramiento y desenmascaramiento de los datagramas IP y gasto de recursos de procesamiento por el hecho de lectura de cabeceras las cuales aumentan su tamaño por motivos del encapsulamiento.

- Una vez descartado uno de los mecanismos de balanceo de carga, se tiene dos posibilidades aún, utilizar balanceo por NAT y utilizar enrutamiento directo. La alternativa de utilizar balanceo por NAT involucra tener un nodo director con una potencia de cálculo superior al utilizado en el método de enrutamiento directo, esto debido a que debe realizar las respectivas traslaciones de direcciones a nivel de servidores reales y nodo director, este proceso involucra tener una capacidad de procesamiento superior ya que el nodo director a mas de realizar las respectivas traslaciones debe recibir los resultados que los servidores reales arrojan y esto entregarlos al cliente.

- Restaría analizar la técnica de balanceo de carga basada en enrutamiento directo, este mecanismo involucra tener un director y los servidores reales en un mismo espacio físico y dentro de la misma red, razón que se acopla al requerimiento de la aplicación a realizarse, también por el hecho de no realizar ningún tipo de encapsulamiento, traslación y reenvío de resultados a los clientes, el servidor director no se va a ver afectado por motivos de sobrecarga en procesamiento. Así como también es importante mencionar que el balanceo de carga va a ser mucho más rápido debido a que el enrutamiento lo realiza a nivel MAC es decir a nivel de capa enlace. El

único problema que se tendría es el hecho de tener la dirección IP virtual configurada en todos los servidores, obteniendo con ello problemas de conflictos al momento de responder requerimientos ARP, pero esto se lo soluciona aplicando ciertos mecanismos explicados en secciones posteriores.

Como se puede observar de acuerdo al análisis realizado el mecanismo de balanceo de carga a utilizarse es el basado en enrutamiento directo, ya que es el que mas se acerca a los requerimientos y necesidades que el *cluster* que se está configurando necesita.

Es importante mencionar que previa la configuración definitiva, se realizaron prueba del caso, mediante el uso del máquinas virtuales utilizando las herramienta de virtualización *VMware*[50] en estas pruebas también se decidió utilizar el mecanismo de balanceo de carga basado en enrutamiento directo, esto debido a las razones expuestas anteriormente.

3.3.1.2 Determinación del tipo de Balanceo de Carga.

Existen varios tipos para realizar el balanceo de carga, estos fueron analizados en el capítulo 2, pero es importante volverlos a mencionar con la finalidad de determinar cual de ellos se acopla a las necesidades y requerimientos buscados para ser utilizados en la configuración del *cluster* de balanceo de carga.

Previo análisis se debe determinar que tipo de *cluster* se va utilizar en términos de hardware, es decir determinar si se va tener un *cluster* homogéneo o heterogéneo.

En el desarrollo del presente proyecto de titulación se va utilizar un *cluster* homogéneo utilizando los mecanismos de virtualización mediante la

---

[50] Ver sitio Web: *www.vmware.com*

herramienta *VMWare* (máquinas virtuales con sistema operativo Linux sobre Windows). Para ello es importante realizar un análisis de los tipos de balanceo de carga para ambientes homogéneos y heterogéneos.

3.3.1.2.1 Análisis del tipo de balanceo de carga en ambientes homogéneos

Para el ambiente de pruebas con un *cluster* homogéneo se pueden utilizar los siguientes tipos de balanceo:

- *Round Robin.* Balanceo mediante el método *FIFO* (*First in First Out)* donde el nodo director envía las peticiones al primer servidor y esto se mantiene así hasta llegar al último, luego del cual se regresa al primero.

- Servidor con menos conexiones activas. Proceso caracterizado por tener un mecanismo continuo de consulta acerca de cuántas conexiones activas poseen los servidores que están atendiendo los requerimientos del cliente, dependiendo de esta consulta el nodo director direcciona las peticiones a los servidores con menos cargas.

Se debe realizar el análisis respectivo para proceder a utilizar uno de ellos, en caso de *Round Robin*, es el tipo más sencillo para realizar el balanceo de carga, su funcionamiento por el hecho de tener un *cluster* homogéneo es bueno, pero se corre el riesgo de que toda la carga pesada vaya a un mismo servidor, mientras que el resto de servidores reciben carga mas liviana, existiendo de esta manera un mecanismo injusto de repartición de carga.

En caso de utilizar el mecanismo de servidor con menos conexiones activas es el mecanismo que posee un mejor comportamiento en ambientes homogéneo, ya que el nodo director sondea constantemente cuantas conexiones activas poseen los servidores reales y según ello distribuye la carga para procesar un determinado trabajo. Por el hecho de tener un procesamiento similar, el número de conexiones activas y en espera van a ser equilibradas y con ello se va tener un

adecuado funcionamiento del cada uno de los servidores reales, logrando con ello tener un rendimiento similar entre ellos.

Luego del análisis respectivo se determinó que el mecanismo de balanceo de carga basado en menos conexiones activas es el que mejor comportamiento tiene en relación al tipo de balanceo basado en *Round Robin*.

3.3.1.2.2 Análisis del tipo de balanceo de carga en ambientes heterogéneos.

Para el ambiente relacionado a la configuración definitiva utilizando un ambiente heterogéneo se hizo un análisis de los mecanismos que mas se acercan a los requerimientos de la presente configuración:

- *Round Robin* ponderado. Tipo que basa su funcionamiento en el método *Round Robin* normal, pero añade un parámetro adicional a cada uno de los servidores el cual depende de su potencia de cálculo, con lo cual servidores con mayor potencia de calculo recibirán mayores conexiones que los que poseen menor capacidad.

- Servidor con menos conexiones activas ponderado. Técnica que basa su funcionamiento en el método de balanceo con menos conexiones activas, al igual que *Round Robin* ponderado asigna pesos a los servidores basándose en su potencia de cálculo, mediante este mecanismo se asigna una mayor carga a los servidores con menos conexiones activas pero verificando la capacidad de cálculo de éstos, y dependiendo de esta capacidad asignar una mayor o menor carga a los servidores, con lo cual elimina el inconveniente de saturar servidores con pocas conexiones activas, en espera y con poca capacidad de calculo .

- Menos conectado basado en servicio. Su funcionamiento se basa en dirigir todas las peticiones a un mismo servidor hasta lograr que se sobrecargue, luego del cual ejecuta la técnica menos conexiones activas ponderadas

sobre el resto de servidores del *cluster*, es decir se pasan las peticiones al servidor con menos conexiones activas, analizando previamente la potencia de cálculo de éstos, el proceso continua hasta que se satura la capacidad de conexiones.

Este mecanismo se mantiene de manera sucesiva hasta llegar al último servidor en servicio del *cluster.* Esta técnica es muy utilizada cuando se ofrece varios servicios distintos y se quiere especializar a una determinada máquina en un servicio determinado, cabe recalcar que todas las máquinas son capaces de reemplazar a cualquiera de lo servidores activos cuando uno de estos falle

De estos tres mecanismos se va utilizar la técnica basada en servidor con menos conexiones activas ponderado, esta selección se la realizó de acuerdo al siguiente análisis:

El tipo de balanceo *Round Robin* ponderado es el mecanismo más sencillo de distribución de carga existente para ambientes heterogéneos, este tipo de balanceo no fue escogido debido a que presenta el problema de que en escenarios donde exista una gran demanda de acceso a los servicios, como es el caso del presente proyecto, los servidores mas potentes están siempre ocupados, tendiendo a estar constantemente saturados, por ello se debe ocupar aquellos que poseen una capacidad de cálculo menor, haciendo de que estos tiendan también a saturarse, inconvenientes que hacen que el funcionamiento del conjunto de servidores no sea el adecuado.

El tipo de balanceo basado en menos conectado basado en servicio y servidor con menos conexiones activas ponderado básicamente poseen el mismo funcionamiento, la diferencia radica únicamente en que el tipo de balanceo menos conectado basado en servicio se caracteriza por saturar a cada servidor con conexiones generadas desde el nodo director y luego del cual aplica la técnica servidor con menos conexiones activas ponderado para determinar cual es el

siguiente servidor a enrutar los trabajos asignados por el nodo director y procede a saturarlo, como se puede observar cualquiera de ellos podría ser útil, es por ello que la técnica de balanceo servidor con menos conexiones activas ponderado podría ser la mas adecuada.

3.3.1.3 Determinación de la herramienta de instalación de LVS.

Es importante aclarar que existen disponibles en el Internet varias herramientas libres de instalación, a continuación se hace un análisis de algunas de ellas que son más conocidas y utilizadas:

- *Ipvsadm.*[51] Herramienta utilizada para configurar los servidores virtuales utilizando LVS, el mecanismo utilizado para ello es mediante el intérprete de comandos o mediante un *script* en el que se detallan un conjunto de órdenes a ejecutarse. *Ipvsadm* permite añadir y quitar servicios y servidores, asignar pesos a servidores, mecanismos de planificación; entre otros.

- *Piranha.*[52] Herramienta propietaria a *RedHat.Inc*, la cual incluye mecanismos para garantizar alta disponibilidad y balanceo de carga de un determinado sistema (trae el parche *IPVS* para el *kernel* en caso de no poseerlo). Presenta para ello una herramienta gráfica de configuración.
  Para al alta disponibilidad utiliza diferentes demonios propietarios los cuales se encuentran constantemente monitoreando al nodo director y en caso de falla de este enrutar al nodo de respaldo.

  Para el caso de balanceo de carga utiliza LVS con diferentes demonios propietarios administrados con la herramienta *Ipvsadm*, mediante la cual permite sacar las ventajas que esta herramienta provee.

---

51 Ver Capítulo 2 ítem 2.2.2.7.1

52 Ver Capítulo 2 ítem 2.2.2.7.2

- *Ultramonkey.*[53] Herramienta de uso libre, la cual permite configurar un *cluster* de alta disponibilidad y balanceo de carga, permite tener diferentes escenarios de configuración, para la alta disponibilidad se basa en los demonios *mon, heartbeat, fake*. Mientras que para garantizar la alta disponibilidad hace uso de la herramienta *Ipvsadm*.

Luego de un análisis rápido de estas herramientas de configuración, se debe escoger una de ellas para poder realizar la respectiva configuración. En el caso se *Ipvsadm* es una herramienta utilizada para configurar LVS y con ellos el balanceo de carga. Esta herramienta no posee una interfaz gráfica para el proceso de configuración, siendo esta una de sus desventajas, pero permite realizar un sin número de actividades muy útiles relacionadas a la configuración de LVS que ya se mencionaron en el ítem respectivo. Cabe mencionar que esta herramienta es utilizada por las otras para términos de configuración de lo que respecta a balanceo de carga.

Con relación a las herramientas *piranha* y *ultramonkey*, son herramientas mas completas que *Ipvsad*m ya que agregan el módulo relacionado a la alta disponibilidad, en cuanto al balanceo de carga utilizan algunos demonios propietarios y de uso libre respectivamente, pero basan su configuración principal en *Ipvsadm*. Es importante mencionar que *ultramonkey* no ofrece un soporte adecuado al usuario.

Piranha es una herramienta útil y completa pero no se la escogió debido a que no es una herramienta de uso libre y por falta de recursos no se pudo adquirirla, con relación a *ultramonkey* es una herramienta libre y completa que provee diferentes escenarios de configuración a nivel de alta disponibilidad y balanceo de carga, pero no provee un adecuado soporte para configurar un servidor de balanceo de carga utilizando un sistema operativo basado en *Fedora Core 5* con una versión de *kernel* 2.6. *Ultramonkey* en comparación con *Ipvsadm* ofrece un valor

[53] Ver Capítulo 2 ítem 2.2.2.7.3

agregado que es la alta disponibilidad, ya que para el balanceo de carga utiliza a *Ipvsadm* como herramienta de configuración, no existiendo ninguna diferencia en funcionamiento entre la una herramienta y la otra en términos de balanceo de carga ya que utilizan la misma.

Es por ello que se escogió la herramienta *Ipvsadm* para realizar la configuración del *cluster* de balanceo de carga, ya que esta herramienta ofrece un adecuado soporte para configurar un servidor trabajando con el sistema operativo *Fedora Core 5* con una versión de *kernel* 2.6. Además es importante mencionar que la alta disponibilidad que ofrece *ultramonkey* debido a problemas de soporte en términos de versión de *kernel* y sistema operativo no es de utilidad para la presente configuración, es por ello que se la va a realizar con otra herramienta que si ofrece soporte que se explicará en los próximos ítems.

3.3.2 LVS – INSTALACIÓN.

Una vez analizadas todas las alternativas para configurar el *cluster* de balanceo de carga se decidió utilizar las siguientes herramientas y configuraciones:

3.3.2.1 Arquitectura del *cluster* de balanceo de carga.

La arquitectura del *cluster* de balanceo de carga basado en enrutamiento directo y sus parámetros de configuración se detallan en la Figura 3-1.

3.3.2.2 Software.

Como se mencionó anteriormente el sistema operativo utilizado es Linux, hablando más concretamente *Fedora Core* Linux, la cual es una distribución GNU/Linux que fue desarrollada por la comunidad *Fedora*

Fedora Core (también conocida como *Fedora* Linux) es una distribución GNU/Linux desarrollada por la comunidad *Fedora* y promovida por la compañía estadounidense Red Hat.

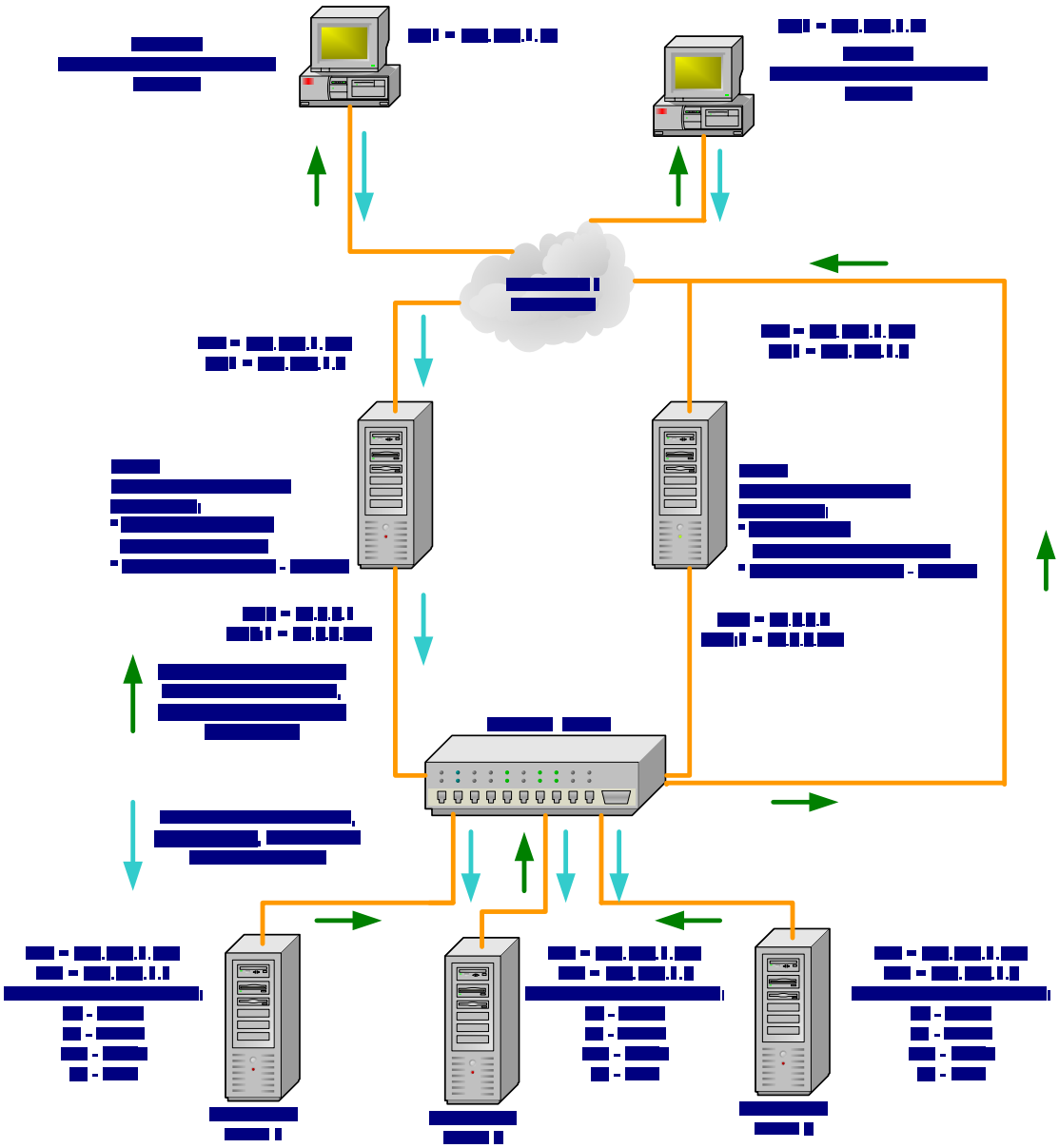


Figura 3-1: Arquitectura del *cluster* de balanceo de carga y alta disponibilidad

El objetivo del proyecto *Fedora* es conseguir un sistema operativo exclusivamente libre, el cual brinde sus herramientas para poder configurar el *cluster* de balanceo de carga y alta disponibilidad. Es por ello que se escogió *Fedora Core* Linux

versión 5, ya que esta versión trae ya incluido en su *kernel* versión 2.6 el paquete de *IPVS* logrando con ello evitar compilar el *kernel* para añadirlo.

A continuación se resume las principales características de *Fedora Core 5*:

- Desarrollado por Proyecto *Fedora.*
- Sistema operativo GNU/Linux.
- Software libre.
- Versión de *kernel:* 2.6.15-1.2054_FC5.
- Fecha de publicación: 20 de marzo del 2006.

Los principales requerimientos son:

- Arquitectura que soporta: Intel x86 (Pentium 4, Pentium Pro, Pentium II, Pentium III), AMD64/EM64T.
- Los requerimentos de memoria para modo texto se necesita mínimo 128 MB o superior, modo gráfico se recomienda 512 MB o superior.
- Los requerimientos de recursos a nivel de disco se necesita: 4 Gigas para el Sistema Operativo.

3.3.2.3 Requerimientos a nivel de *kernel*

Para el correcto funcionamiento y adecuada instalación de LVS, se requiere que el *kernel* utilizado tenga integrado el servicio *IPVS* (*IP Virtual Server),* y poseer configuradas correctamente las interfaces de red instaladas en los servidores.

Existen diferentes versiones de *kernel* disponibles en las diferentes presentaciones de Linux, dependiendo de mencionadas versiones se debe actualizar o no el *kernel* con la finalidad de integrar *IPVS* en su estructura. A

continuación se realiza un pequeño resumen de las versiones de *kernel* existentes y se detalla si requieren o no añadir el paquete *IPVS.*

- Versión de *kernel* 2.6.x. Para todas las versiones del *kernel* 2.6.x viene ya incluido en su estructura el módulo de *IPVS.*

- Versión de *kernel* 2.4.x. Para la versión 2.4.23 y superiores el módulo *IPVS* viene ya incluido en su estructura, para versiones inferiores se debe parchar el *kernel* para añadirlo, para ello se debe utilizar la versión adecuada de acuerdo al tipo de *kernel* que se tenga, ver Tabla 3-6 donde se detallan los tipos de *IPVS* para las diferentes versiones de *kernel* existentes, en caso de querer descargarlas utilizar la siguiente dirección electrónica : *http://www.linuxvirtualserver.org/software/index.html*

- Versión de *kernel* 2.2.x. Para estas versiones se requiere descargar *IPVS* (*http://www.linuxvirtualserver.org/software/index.html*) y parcharlo en el respectivo *kernel.*

Tipo de *IPVS*	**Versión**	**Fecha de Lanzamiento**
IPVS para *kernel* 2.6	1.2.1	24 de diciembre del 2004
IPVS para *kernel* 2.5	1.1.7	05 de julio del 2003
IPVS para *kernel* 2.4	1.0.12	17 de noviembre del2004
IPVS para *kernel* 2.2	1.0.8	14 de mayo del 2001

Tabla 3-6: *IPVS* para diferentes versiones de *Kernel*[5]

Es importante dejar en claro que de acuerdo al mecanismo de balanceo de carga escogido (enrutamiento directo) existen inconvenientes en la versión de *kernel* 2.6, ya que no trae incluido el módulo que evite tener el problema relacionado con conflictos ARP debido a la existencia de la dirección IP virtual configurada en servidores reales y director, es por ello que se debe analizar algunos mecanismos

y escoger uno de ellos para corregir estos inconvenientes, este análisis se va realizar en el ítem de instalación de *Ipvsadm.*

3.3.2.4 Instalación de *Ipvsadm*

La herramienta *Ipvsadm* debe ser parchada en cada uno de los *kernel's* de cada uno de los servidores, sean estos reales o director. Dependiendo de la versión utilizada existe una determinada versión de *Ipvsadm*, a continuación se menciona lo dicho:

- Para versiones de *kernel* 2.6, se debe utilizar *Ipvsadm* versión 1.24 o superior.
- Para versiones de *kernel* 2.4, se debe utilizar *Ipvsadm* versión 1.21.
- Para versiones de *kernel* 2.2, se debe utilizar *Ipvsadm* versión 1.15.

Para poder descargar mencionados parches se lo puede realizar del siguiente url: *http://www.linuxvirtualserver.org/software/ipvs.html#kernel-2.6.*

Una vez descargada la respectiva versión que en este caso va ser la versión de *Ipvsadm-1.24-6.src.rpm.* Se procede a instalarlo con el siguiente comando:

rpm –ivh Ipvsadm-1.24-6.src.rpm

3.3.2.5 Configurando LVS con *Ipvsadm.*

3.3.2.5.1 Descripción de Ipvsadm

Ipvsadm es usado para instalar, mantener o inspeccionar las tablas del Servidor Virtual en el Kernel de linux. LVS puede ser usado para construir servicios de red escalables basados en un *cluster* de dos o más nodos. El nodo activo del *cluster* redirecciona los requerimientos de servicio al conjunto de servidores reales los cuales están prestos atender esos servicios.

Ipvsadm permite trabajar con dos protocolos TCP y UDP, así como tres métodos de balanceo de carga NAT, encapsulamiento IP, enrutamiento directo, y ocho tipos de balanceo de carga (*round robin, round robin* ponderado, servidor con menos conexiones activas, servidor con menos conexiones activas ponderado, Menos conectado basado en servicio, tablas hash por origen y destino Conexiones persistentes).

Ipvsadm para su funcionamiento tiene dos formatos básicos de ejecución:

- **ipvsadm** COMANDO [Protocolo] [dirección IP de servicio virtual][: Puerto] [tipos de balanceo de carga] [Opciones relacionadas a conexiones persistentes]

- **ipvsadm** comando [Protocolo] [dirección IP de servicio virtual] [: Puerto] [dirección IP de servidor real] [: Puerto] [mecanismo de balanceo de carga] [opciones de peso]

El primer formato de ejecución sirve para manipular el servicio virtual, asignar el mecanismo de balanceo de carga, además permite asignar algunas opciones relacionadas a servicios persistentes que pueden ser asignados.

El Segundo formato es utilizado para manipular los servidores reales, asignar un mecanismo adecuado de balanceo de carga, así como asignar pesos a sus servidores.

La descripción de las diferentes opciones de configuración utilizadas en los dos formatos de ejecución se detallan en el anexo A.

3.3.2.5.2 Configurando Ipvsadm.

Para poner en funcionamiento *Ipvsadm* se decidió realizar una aplicación la cual permite configurar tanto para servidores reales, como el nodo director, y modificar

algunos parámetros propios del sistema operativo para poder tener en funcionamiento todo el sistema.

- Algunas consideraciones para funcionamiento de *ipvsadm.* Antes de empezar a configurar, se debe tener en cuenta algunas consideraciones como las mencionadas a continuación:

  - Si se está utilizando el mecanismo de balanceo de carga basado en enrutamiento directo, donde los paquetes que arriban a los servidores reales tienen como dirección destino la dirección IP virtual VIP, es necesario que la máscara de red de esta interfaz sea 255.255.255.255, debido que la interfaz configurada como virtual debe aceptar únicamente paquetes cuyo destino es la dirección IP virtual, mas no otros paquetes que lleguen en caso de tener un rango de direcciones configuradas con mascara 255.255.255.0 por ejemplo, donde la dirección VIP debe aceptar paquetes provenientes de todas las direcciones IP comprendidas en ese rango, lo cual generaría inconvenientes debido a que debe procesar paquetes que no deben ser recibidos, logrando con ello tener deficiencias en su funcionamiento.

- Configuración de LVS mediante interfaz gráfica de usuario. Para el proceso de configuración del *cluster* de balanceo de carga se utilizó la herramienta *Qt designer*[54], herramienta desarrollada usando C++, y que permite diseñar e implementar interfaces con un conjunto de herramientas GUI, cuyo código puede ser portable a varios sistemas operativos como Windows, Linux, Mac OS X; entre otros.

  Se desarrolló una aplicación la cual es útil para la configuración de parámetros de funcionamiento de servidores reales y nodo director,

[54] Visitar sitio Web: http://www.trolltech.com/qt.

dejando claro que dichos parámetros de configuración se basan en la utilización de la herramienta *ipvsadm* la cual es útil para poner en operación un sistema de balanceo de carga mediante LVS.
Para la compilación de la interfaz gráfica, se utilizó el comando *qmake* y *make* estructurados en un *script* denominado *scriptqt* al cual luego de ejecutarlo debe presentar la interfaz gráfica de configuración.

La interfaz gráfica se denomina Administrador – LVS la cual esta estructurada de 4 ítems principales: Configuración de servicios, interfaces de red, monitoreo y estado de LVS

En la parte correspondientes a configuración de servicios se presenta la Figura 3-2, donde se puede observar que existen varias pestañas de configuración. La primera permite especificar parámetros de configuración de los servidores reales.

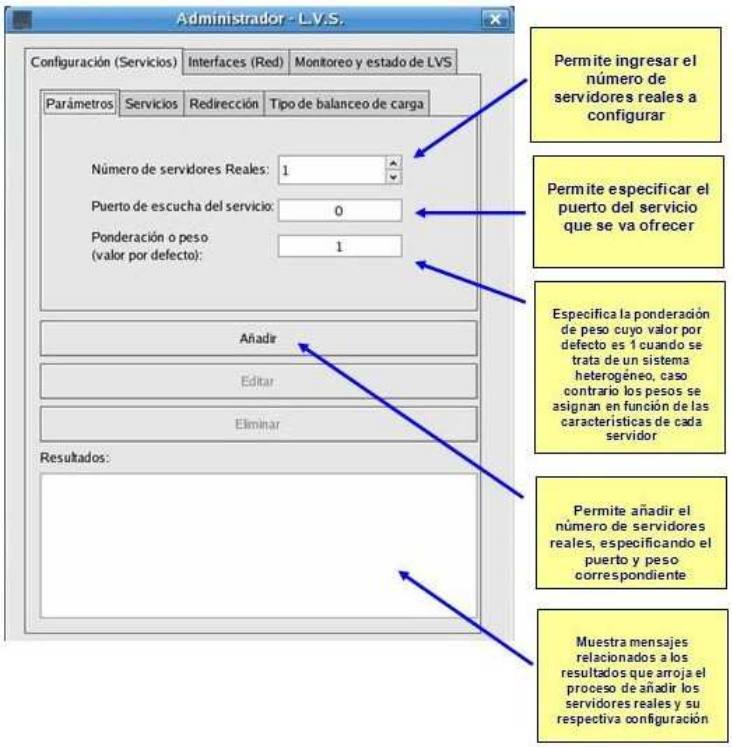


Figura 3-2: Ingreso de Parámetros de Configuración

La aplicación presenta algunas restricciones, en la Figura 3-3 se muestra un mensaje de error el cual indica que necesariamente se debe ingresar el puerto correspondiente al servicio que se este ofreciendo, el cual no debe ser 0 y debe estar comprendido entre los valores 1 – 65535.

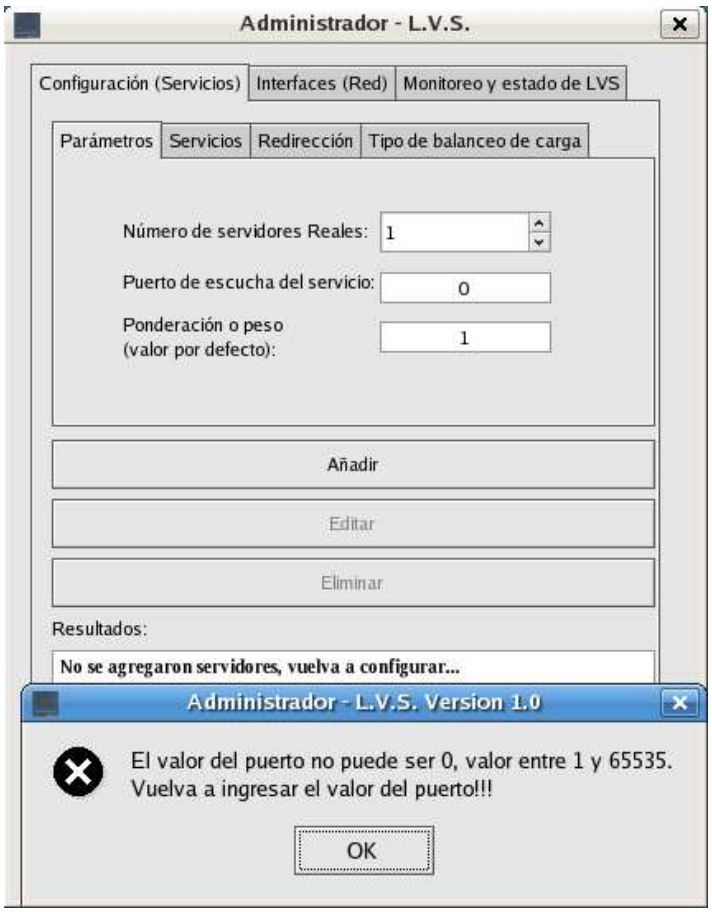


Figura 3-3: Errores cuando no se especifica el puerto.

De igual manera se presenta un mensaje de error cuando el número de servidores reales ingresados es igual a 1, la Figura 3-4 presenta un mensaje donde especifica que no es coherente tener un solo servidor real, ya que la configuración del cluster por naturaleza debe tener como mínimo 2 servidores reales para el proceso de balanceo de carga.

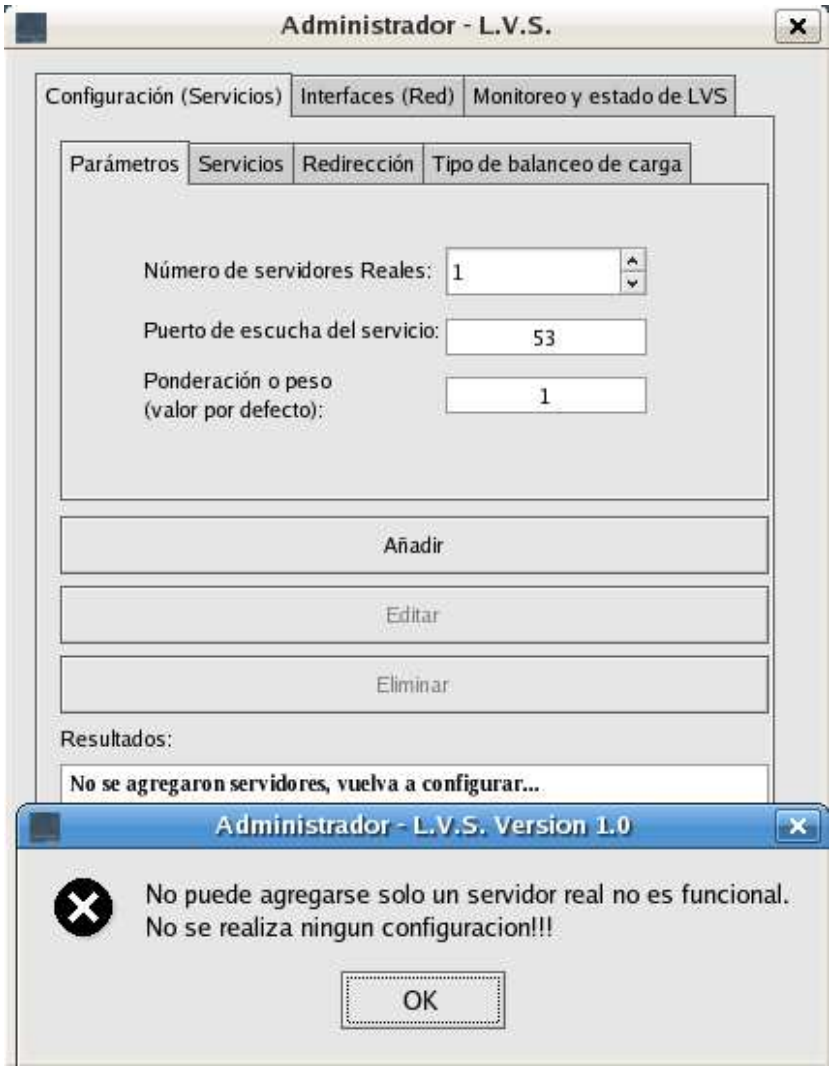


Figura 3-4: Error producido al ingresar un solo servidor real

Una vez ingresado los valores de configuración de manera adecuada, es decir especificando el número adecuado de servidores reales, el puerto asociado al servicio que se esta ofreciendo y la ponderación de peso que por defecto es igual a 1 cuando se trata de un cluster homogéneo, se presenta la Figura 3-5, la cual indica en el recuadro de resultados que la operación es satisfactoria.

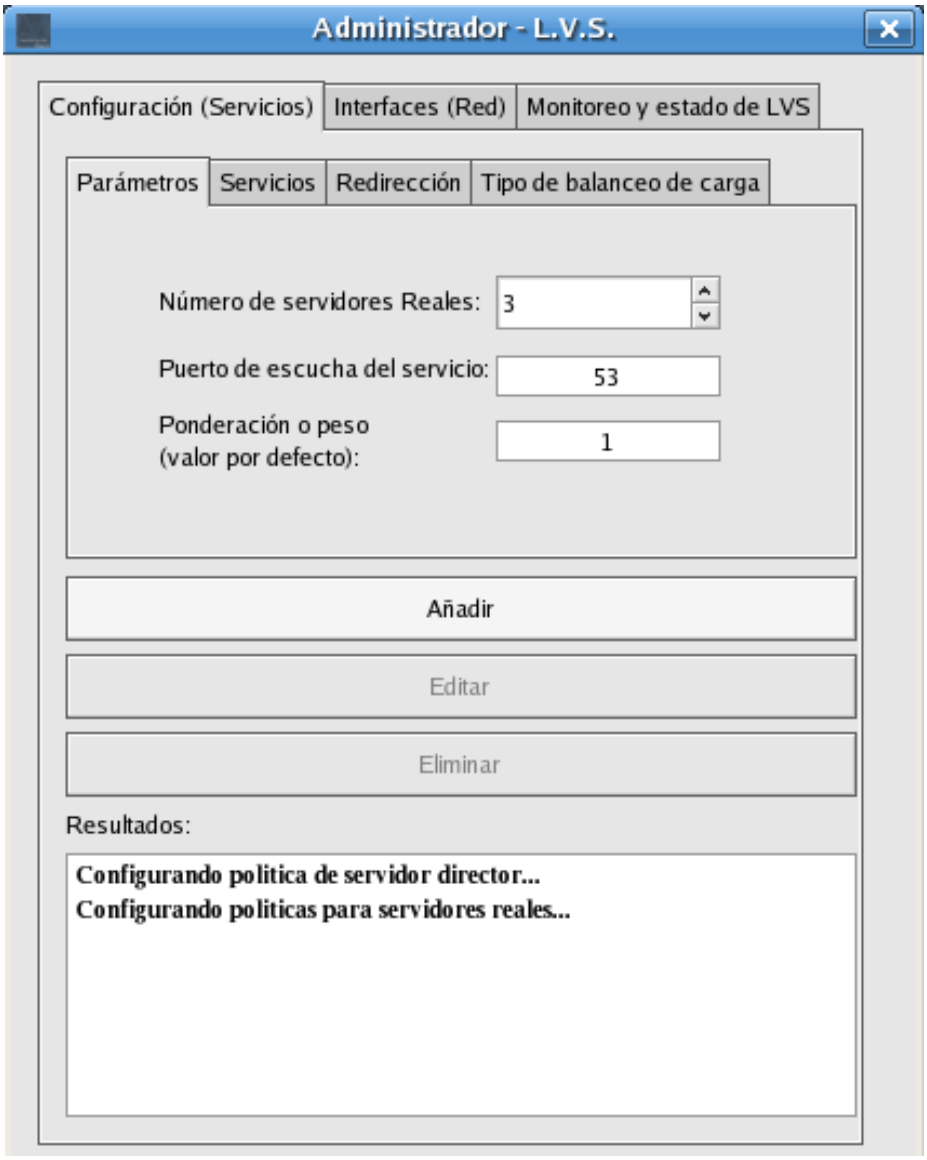


Figura 3-5: Parámetros agregados correctamente

Luego se procede a configurar el tipo de servicio que el sistema va ofrecer, para la presente configuración se va a utilizar servicios TCP y UDP. La Figura 3-6 muestra lo dicho.

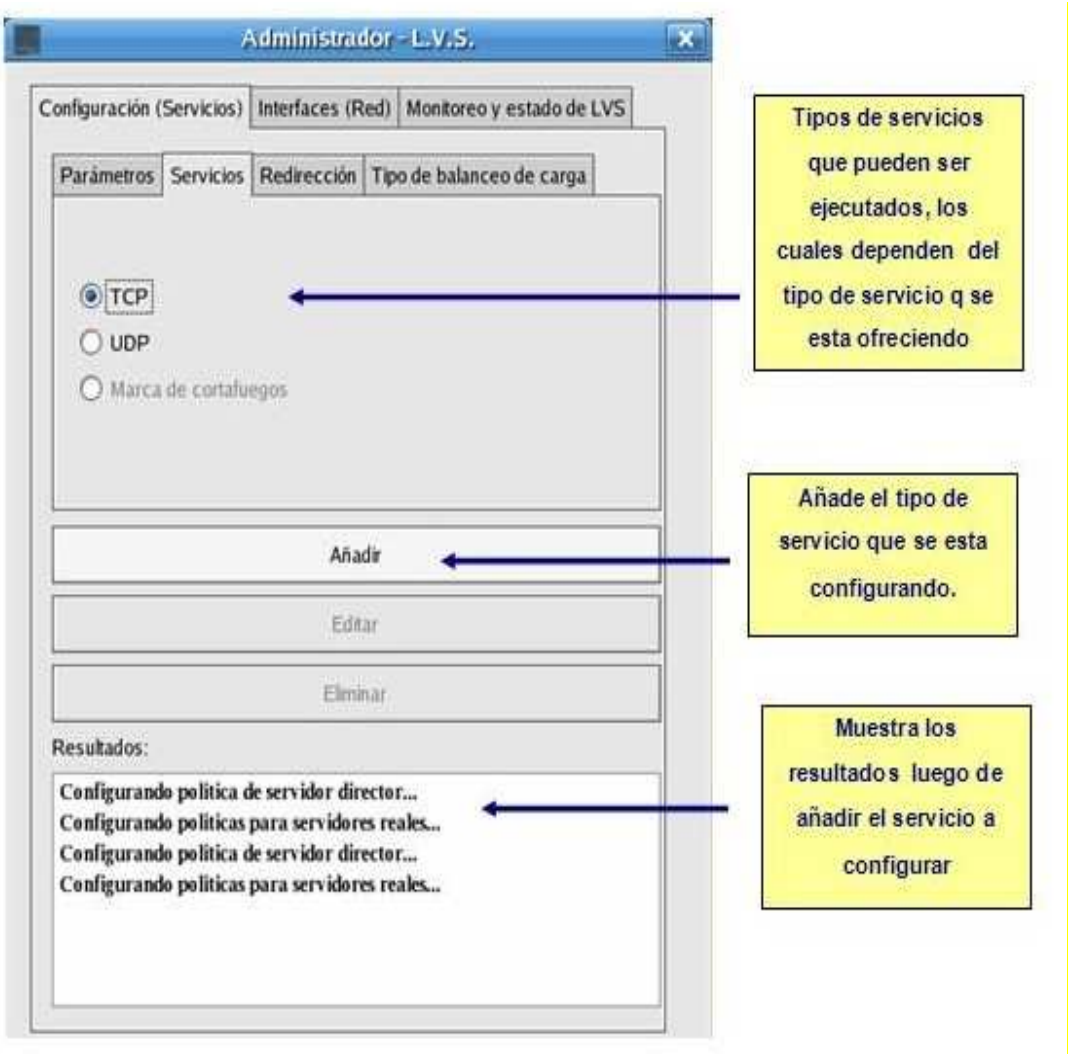


Figura 3-6: Configuración de Servicios

La redirección es un parámetro muy importante para el proceso de configuración de balanceo de carga, para el presente proyecto únicamente se desarrolló la opción de enrutamiento directo, la Figura 3-7 muestra lo comentado.

Por último se debe escoger el mecanismo de balanceo de carga a utilizarse, la Figura 3-8 muestra los distintos mecanismos existentes.

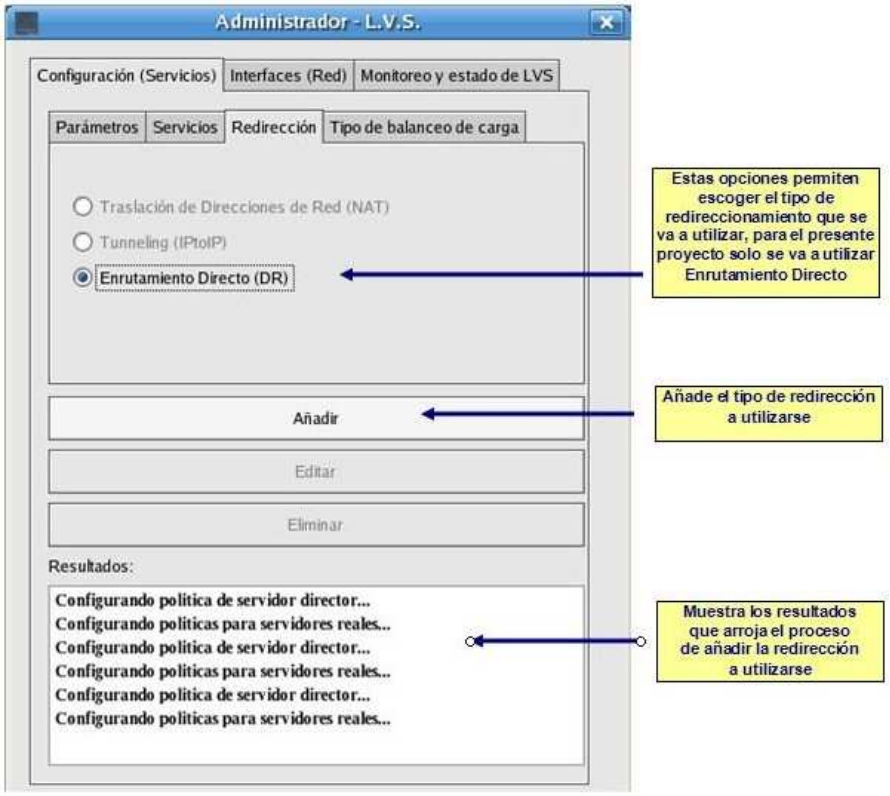


Figura 3-7: Configuración de mecanismo de redirección

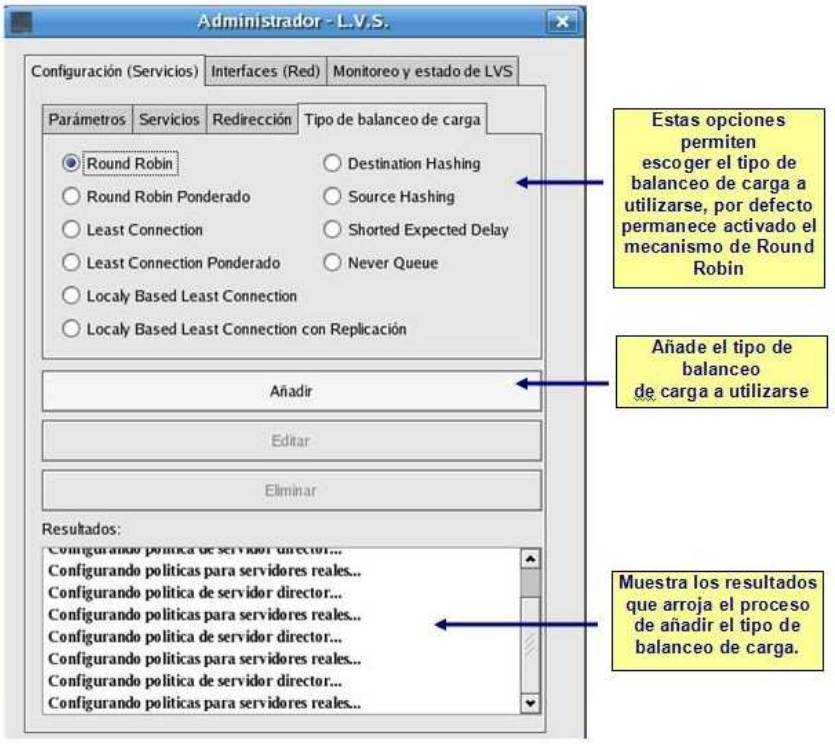


Figura 3-8: Configuración del tipo de balanceo de carga

Configuración de Interfaces de Red Virtuales. - Otro de los ítems de configuración principal es el de configuración de la dirección IP virtual, la cual se va a configurar en el nodo director, esta dirección va a ser la encargada de brindar el servicio a los clientes mediante el nodo director y a través de este los servidores reales, como se muestra en la Figura 3-9.

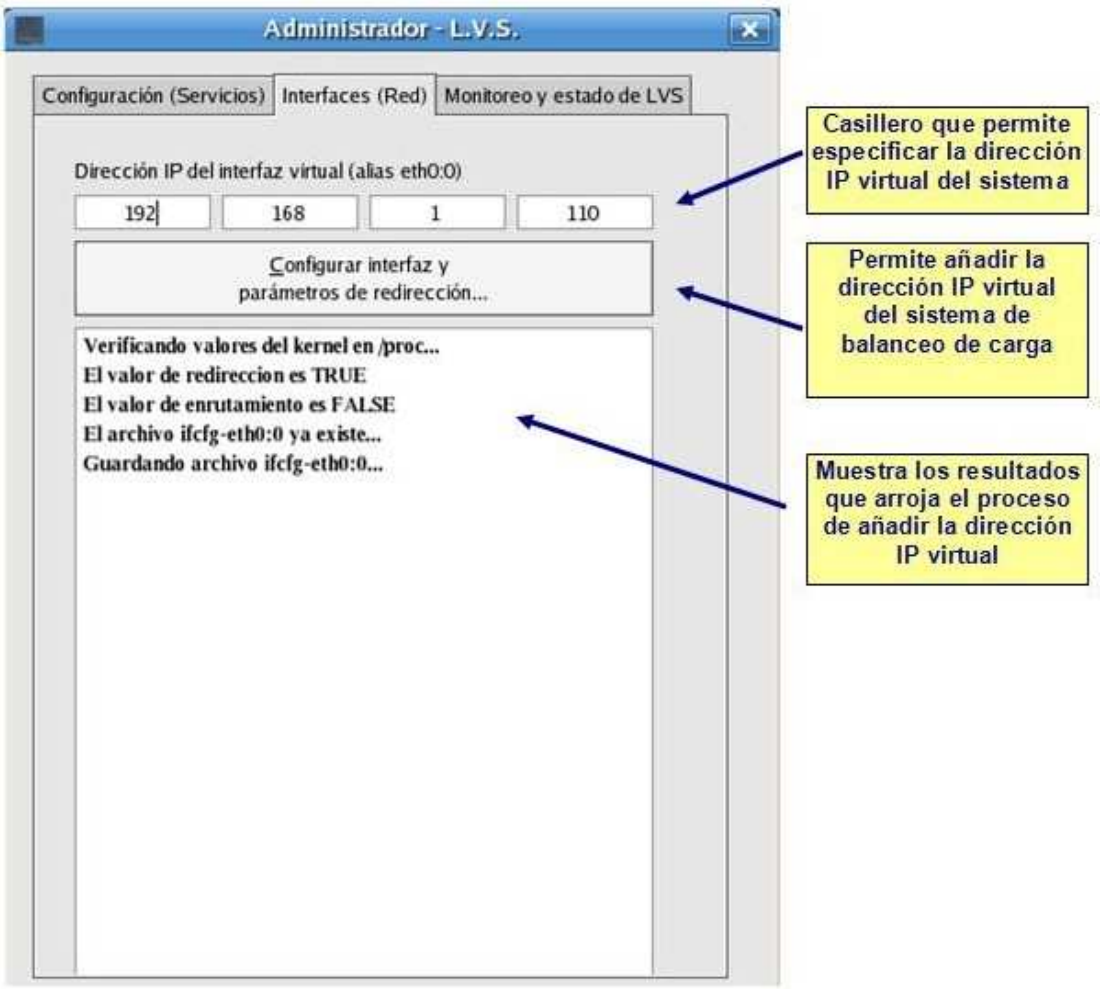


Figura 3-9:Configuración de interfaz de red virtual

Una vez especificada la dirección IP virtual, aparece un mensaje donde pregunta si desea guardar o no dicha dirección IP en el archivo ifcfg-eth0:0, la Figura 3-10 muestra lo dicho.

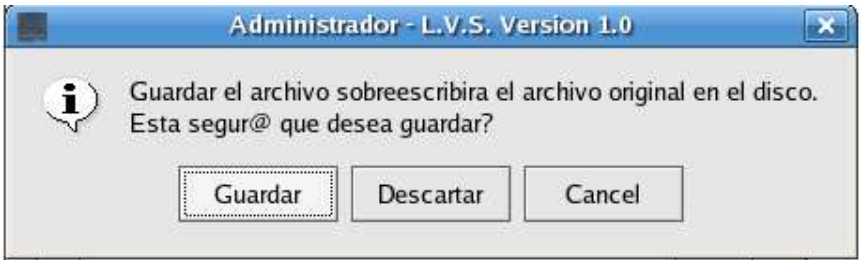


Figura 3-10: Confirmación para guardar permanentemente la dirección IP virtual

Monitoreo y estado de LVS.- En esta pestaña permite monitorear el estado de LVS, mediante la cual muestra información relacionada al estado de las conexiones existentes con los servidores reales y los servicios que estos se encuentran ofreciendo.

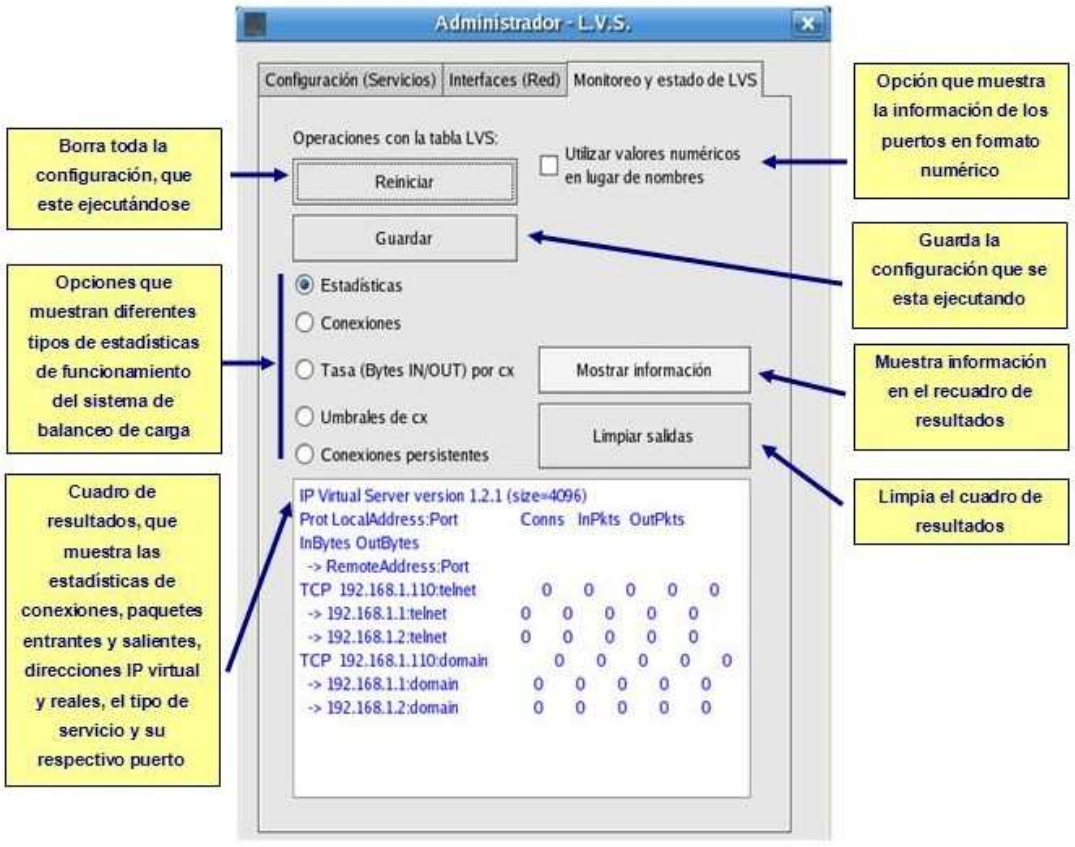


Figura 3-11: Monitoreo y estado de LVS

3.3.2.5.3 Problemas de ARP en balanceo de carga basado en enrutamiento directo.

En el *cluster* LVS/Enrutamiento Directo, la dirección IP virtual VIP es compartida por el nodo director y todos los servidores reales. Para hacer el trabajo de balanceo de carga, el nodo director debe hacer un *broadcast* de la dirección virtual para aceptar entrada de paquetes para el servicio virtual, los servidores virtuales solo utilizan la dirección virtual para procesar los paquetes de manera local.

El problema de ARP surge cuando los servidores reales tienen una de sus interfaces conectadas a la red donde LVS/DR recibe los paquetes destinados para VIP. Si no se deshabilita ARP para la dirección virtual en los servidores reales y nodo director se va dar una respuesta simultanea de la dirección IP virtual, entonces el *router* puede enviar requisitos de manera directa a los servidores reales sin necesidad de hacerlo a través del nodo director, haciendo con esto que no se cumpla el principio de balanceo de carga.

Por ello los servidores reales no deben tener ninguna interfaz conectada a la red que el nodo director recibe los paquetes para la dirección IP virtual, pero si debe tener el mecanismo para responder directamente al cliente mediante el router, sin necesidad de hacerlo a través del nodo director.

En Linux existen varios mecanismos para deshabilitar ARP, de los cuales se va a realizar un breve análisis de estos:

- Usando arptables para deshabilitar ARP. *Arptables* es usado para establecer, mantener e inspeccionar en el *kernel* de Linux las tablas ARP y reglas de filtrado de paquetes. Varias tablas diferentes pueden ser definidas, cada tabla esta constituida por cadenas. Cada cadena es una lista de reglas que pueden emparejar un conjunto de paquetes. Cada regla determina que hacer con un paquete que empareja.

Para deshabilitar ARP se debe omitir los pedidos entrantes de ARP para las direcciones virtuales de los servidores reales que vienen especificadas como destino, así como se debe tener cuidado con los pedidos salientes de ARP desde las direcciones virtuales, debido a que los servidores reales envían al cliente paquetes de respuesta con origen la IP virtual y generaría requerimientos ARP para dicha dirección virtual configurada, haciendo que los demás servidores contesten a estos requerimientos ARP y no se podría enviar la respuesta al cliente debido a que el paquete enviado por el servidor real no llegaría al *router* de salida, para ello se debe cambiar el origen del paquete de respuesta por la dirección real del servidor real para que sea esta la que emita la respuesta.

- Usando arp_announce/arp_ignore para deshabilitar ARP. Para deshabilitar ARP en IP virtuales de los servidores reales, se necesita modificar los parámetros para *arp_announce/arp_ignore* en las interfaces de red VIP.

- Ocultando la interfaz para deshabilitar ARP. La bandera *hidden* en una interfaz es utilizada para ocultar la interfaz para evitar responder a requerimientos ARP. Cuando esta bandera es habilitada en una interfaz, ninguna dirección IP configurada en la interfaz oculta debería participar en el ARP.

  La bandera *hidden* está habilitada en el *kernel* de Linux versión 2.2 empreñando desde la serie 2.2.14, pero para las versiones de *kernel* 2.4 y 2.6, se necesita aplicar un parche al *kernel* de Linux para habilitarla, el cual se lo puede obtener en el siguiente sitio Web: *http://www.ssi.bg/~ja/#hidden.*

- Usando redirecciones para deshabilitar ARP. Técnica encargada de modificar las características de *iptables/ipchains* para redireccionar los paquetes a una dirección IP y puerto local de los servidores reales.

Para mayor información de estos mecanismos, referirse al siguiente sitio Web: *http://kb.linuxvirtualserver.org/wiki/ARP_Issues_in_LVS/DR_and_LVS/TUN_Clusters*

Luego de realizar el análisis anterior se puede observar que la función de cada método especificado para solucionar los problemas de ARP que ocasiona el tener la misma interfaz virtual configurada en los servidores reales y nodo director es el mismo, solo que se lo hace de diferente manera, pero el resultado de estos es el mismo, llegando a la conclusión de que se puede utilizar cualquiera de ellos, para este caso se escogió el mecanismo de ocultar las interfaces de los servidores reales.

3.3.2.5.4 Ocultando la interfaz para deshabilitar ARP

Como la versión del *kernel* que se está utilizando es la 2.6, para utilizar este método se debe ejecutar un parche para ocultar las interfaces, en este proceso se debe utilizar el parche *hidden-2.6.1-1*, el cual se debe descargarlo del siguiente sitio Web: *http://www.ssi.bg/~ja/#hidden*.

Para poder aplicar el parche se debe recompilar el *kernel*, este es un proceso que se debe tener mucho cuidado ya que se puede comprometer el funcionamiento del sistema operativo. El proceso se detalla a continuación:

- Instalación de las fuentes del *kernel.* Para la instalación del kernel de Linux se debe descargar los binarios *Linux-version.tar.gz* o *linux.version.tar.bz2* desde las siguientes fuentes fuente:

  - CD-ROM: *kernel-source-version.i386.rpm*
  - *www.kernel.org*
  - *www.linux.org*

  Estos archivos deben descomprimirse en el directorio de fuentes: */usr/src/linux-version.* Para asegurarse que las opciones de configuración

se preservaron de la persona quien creo el rpm o el tar.gz se debe ejecutar: *make mrproper*.

Al tener la versión adecuada instalar el RPM:

- *rpm -ivh kernel-version.src.rpm,* (para este caso la versión de *kernel* utilizada es 2.6.15-1.2054_FC5)

Los contenidos del RPM se escriben en */usr/src/redhat/SOURCES* y */usr/src/redhat/SPECS*; luego de ello se debe preparar las fuentes del *kernel* para ello se debe ejecutar los siguientes comandos:

- cd */usr/src/redhat/SPECS*
- rpmbuild -bp -target $(arch) *kernel*.spec

Como resultado el árbol de la fuente del *kernel* se ubica en */usr/src/redhat/BUlLD/kernel-version* sin embargo es más práctico mover el directorio resultante al */usr/src* (opcional no necesario):

- *cd /usr/src/redhat/BUILD/kernel-version*
- *mv linux-version /usr/src* (version de linux creada: linux-2.6.15-i686).
- *cd /usr/src*
- *In -s ./linux-version/ linux*

- Compilación de *kernel.* Se debe ubicar en el directorio *cd /usr/src/linux*. Las configuraciones para un *kernel* específico deben ubicarse en el directorio *configs*
  - *cp configs/<archivo de configuración deseado>.config*.
  - Se ejecuta *make mrproper* para preservar la configuración del archivo .config seleccionado
  - Por último se ejecuta *make oldconfig* para importar la información de configuración en la nueva configuración de *kernel.*

- Aplicación de Parches para el *kernel.* Se debe parchar el *kernel* para modificar el código fuente del *kernel* para incluir el soporte de nuevo hardware o cambiar la funcionalidad y típicamente se publican estas actualizaciones como parches. Un archivo de parche solo lista las diferencias entre el antiguo y el nuevo código y no contiene código sin modificar. El formato de los archivos de parches universal es *diff.*

  Para aplicar el parche se utiliza el siguiente comando:

  - *patch –p0 < /root/hidden-2.6.12.1.diff*. La opción *–p0* es utilizada para ignorar el directorio base del árbol (asegura que el parche se aplique en el directorio */usr/src/linux* el cual fue creado originalmente desde un árbol en */root/linux).* Para eliminar o regresar al estado original antes de aplicar un parche se utiliza la opción -R.

  Luego de aplicar los parches es indispensable configurar el *kernel:*

  - *make menuconfig* y todas las configuraciones se almacenan en el archivo *.config* que se encuentra en el directorio */usr/src/linux-version.*
  - *vi Makefile*

  Después de realizar la configuración se desearía limpiar el árbol de instalación con *make clean* (elimina todos los archivos temporales *.o y *.a). Una vez eliminados los archivos temporales se vuelven a crear las dependencias de los módulos: *make dep.* Luego de esto se procede a compilar el *kernel* con *make bzImage*. Si se configuraron ciertas partes del *kernel* como módulos es necesario ejecutar el comando *make modules.*

- Configuración del nuevo *kernel.* Se debe ubicar en el directorio *cd /usr/src/Linux*, posteriormente se copia la imagen existente en el archivo *.config* al directorio */boot.* Por conveniencia se renombra la imagen del *kernel* para incluir la versión completa.

- *cp .config /boot/config-2.6.15-LVS*

Es una buena idea copiar y renombrar los archivos *System.map*, el cual contiene la ubicación de todos los símbolos o variables globales y rutinas del *kernel*.

- *cp system.map /boot/system.map-2.6.15-LVS*

Se debe copiar también la imagen del *kernel* al directorio */boot.*

- *cp arch/i386/boot/bzImage /boot/vmlinuz-2.6.15-LVS*

- Instalación de nuevos módulos del *kernel*. Para instalar los módulos se debe ejecutar el comando:

  - *make modules _install.*

- Imagen con módulos a cargarse al arranque del sistema. Se utiliza el *InitRD* (*initial root disk*) si se necesita cargar módulos para acceder al sistema de archivos raíz. *InitRD* se crea con el comando *mkinitrd* (*Fedora/Red Hat*).

  - *cd /boot/*
  - *mkinitrd initrd-2.6.15-LVS.img 2.5.15-LVS*

- El último paso es editar el archivo de configuración LILO o GRUB y añadir el nuevo *kernel.* Una vez realizado este proceso se debe reiniciar el sistema, seleccionar el nuevo *kernel* y comprobar su estado.

  - *vi grub/menu.lst*
  - *reboot*

3.3.3 *HA OSCAR* – INSTALACIÓN.

HA OSCAR ha desarrollado un paquete de instalación por medio de una interfaz GUI. Primeramente se debe descargar la última versión de *HA OSCAR*, esto se lo puede hacer desde el sitio Web: *http://cenit.latech.edu/oscar*, luego del cual se procede a instalarlo.

El ayudante de instalación presenta varias páginas, para lo cual se debe ingresar como usuario "*root*" antes de iniciar la instalación. La interfaz directiva es la interfaz de red privada para el nodo primario, la cual es normalmente "*eth0*". El ayudante de instalación de HA-OSCAR permite encaminar al usuario a través de un completo proceso de instalación que normalmente consiste en:

1. Instalar los paquetes HA-OSCAR.
2. Construir la imagen del Servidor de *Standby* (clonando el servidor primario).
3. Configuración inicial de Servidor de *Standby.*
4. Configurar la red y crear la imagen de inicio en el Servidor de *Standby.*
5. Completar la instalación

3.3.3.1 Paquete de Instalación HA-OSCAR

El primer paso es instalar todos los paquetes para configura *HA-OSCAR,* el ayudante instalará todos los paquetes requeridos para el servidor Cluster *HA-OSCAR* y prepara el ambiente para los siguientes pasos.

Figura 3-12: Instalación de HA OSCAR

3.3.3.2 Clonando la Imagen para el Servidor Standby.

Cuando el usuario selecciona la opción "*Building Image for Standby server*", el ayudante abre otra ventana donde se puede definir el nombre de la imagen del servidor. Normalmente, se puede elegir el valor por defecto y luego se debe presionar el botón "*Fetch image*" para obtener una imagen del servidor *standby*. Una vez terminada la clonación se debe presionar el botón "*close*" para ir al siguiente paso, en la Figura 3-13 se puede observar como seguir el paso antes descrito.

Figura 3-13: Especificación del nombre de imagen a crear

3.3.3.3 Configuración del Servidor *Standby*

Éste paso requiere que se ingrese un alias a la dirección IP pública, al iniciar éste proceso HA-OSCAR muestra la página "*the Standby Server initial network Configuration screen*" mostrado en la Figura 3-14. Normalmente se utiliza esta dirección IP pública como un punto de entrada virtual para acceder al nodo de cabecera. Cuando ocurre algún error, el servidor *standby* debería tener asignado la dirección IP pública para que los usuarios puedan seguir accediendo al cluster como si nada hubiera ocurrido. El procedimiento para seguir dicho paso es el siguiente:

1. Ingresar el alias de la IP pública para el Interfaz que se esté utilizando. Cuando ocurre algún error el servidor *standby* clonará automáticamente el grupo de IP públicas en la interfaz de red designada.

2. Determinar si la dirección IP local *standby* no ha sido reservada para otro nodo. Si dicha dirección IP se encuentra ocupada, se debe seleccionar una nueva dirección.

3. No es necesario cambiar los otro dos ítems.

4. Cuando todo se encuentre bien configurado, se debe hacer clic en el botón "*Add Standby Server*".

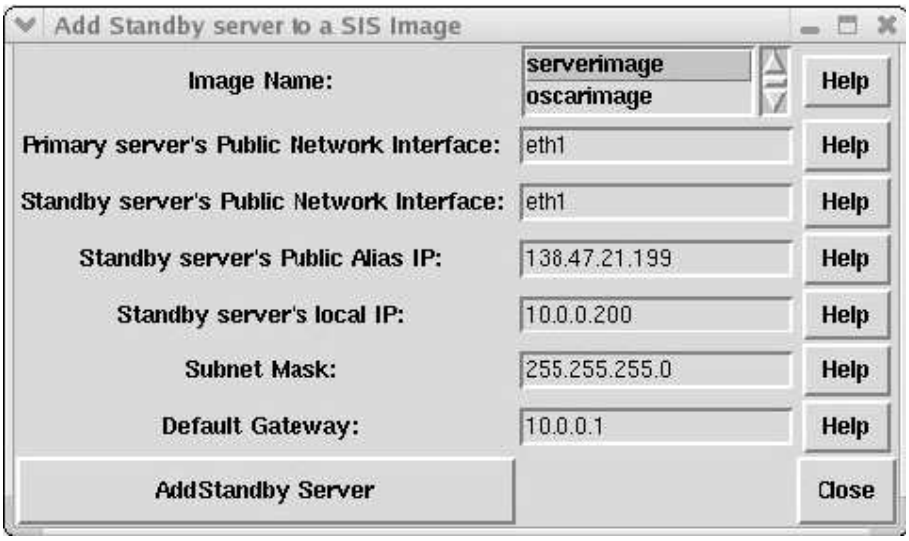


Figura 3-14: Especificación de direcciones reales y virtuales para el servidor *standby*

Al igual que el paso anterior para poder seguir con la instalación se debe hacer clic en el botón cerrar.

3.3.3.4 Recuperar la dirección MAC del servidor *standby* y construir la imagen en el disco local

Es importante poner atención a los procedimientos siguientes para recuperar la dirección MAC del servidor *standby* para el proceso de PXE booting, esto antes de construir sus imágenes en el disco duro local.

Uno de los interfaces de red del servidor *standby* eth0, es típicamente conectada a la red LAN privada y reservada para transmitir un *broadcast* de su dirección MAC durante el inicio de la red.

Cuando el servidor primario se encuentra listo para construir la imagen del servidor *standby*, empieza clonando sus imágenes con las direcciones completas.

Consecuentemente, el servidor *standby* obtendrá la imagen por la red, y la inicializa el servidor *standby* vía PXE (o diskete) desde el servidor principal o un servidor de imagen opcional en este sistema de archivos local. Cuando el servidor *standby* es clonado satisfactoriamente, éste debería ser reiniciado desde el disco duro donde se instaló la imagen. Esto marca la instalación completa del servidor *standby*.

A continuación se especifican los pasos a seguir para asignar la dirección MAC de reserva para el servidor *standby*, para lo cual HA-OSCAR despliega una página llamada "*The standby server MAC address configuration screen*" mostrada en la Figura 3-15.

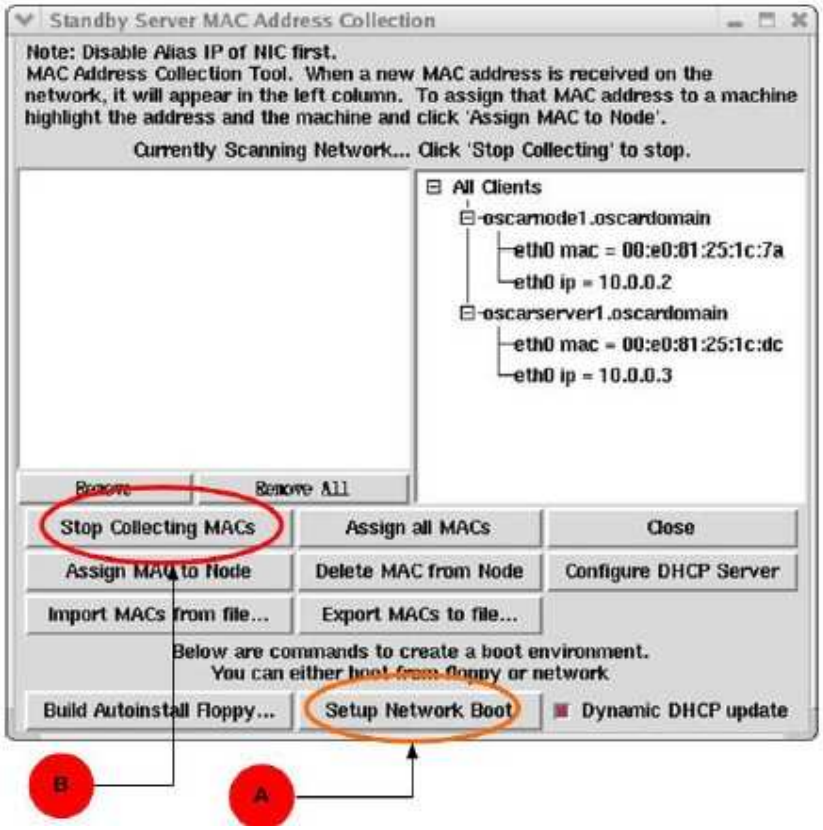


Figura 3-15: Configuración de la dirección MAC del servidor de standby

El procedimiento a seguir es el siguiente:

a) Hacer clic en el botón "*Setup Network Boot*"

b) Hacer clic en el botón "*Start Collect MAC address*" para indicar al servidor primario el conjunto de direcciones MAC de reserva de los servidores.

c) Conectar el servidor Terminal de reserva, se debe configurar la secuencia de inicio para comenzar con el inicio de la red, y reiniciar el sistema de reserva. Asegurarse de que la interfaz eth0 del servidor *standby* este conectado al switch local en donde el servidor primario donde se ejecuta el demonio PXE está escuchado en petición de inicio emitida desde el servidor *standby*. En caso de no cumplir con esto, el servidor primario no esta habilitado para recolectar las direcciones MAC del servidor *standby.*

d) Seleccionar en la parte superior izquierda de la ventana abierta las direcciones MAC de los servidores de reserva.

e) Asignar la dirección MAC a la interfaz de red del servidor de reserva.

f) Hacer clic en el botón "*Configure DHCP Server*".

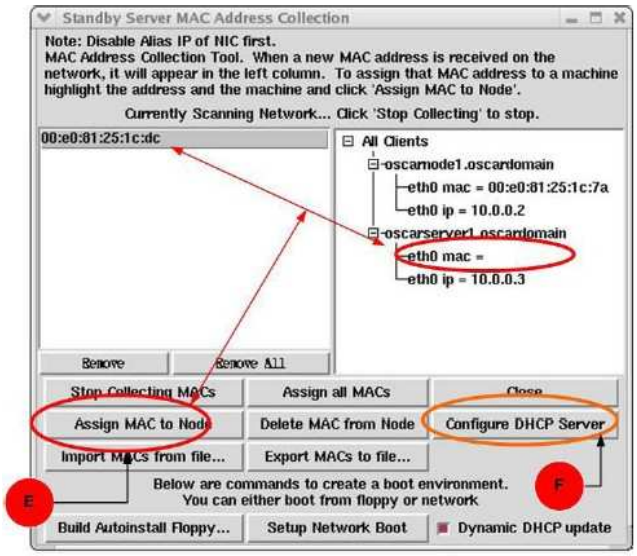


Figura 3-16: Recolección de direcciones MAC del servidor de standby

g) Retornar al servidor *standby* y reiniciarlo, este debe empezar a construir su imagen local.

h) Mientras la imagen del servidor de reserva es completada, se debe asegurar que el inicio del sistema se realice primero por medio del disco local, y luego se debe reiniciar el sistema.

3.3.4 *HA OSCAR* – CONFIGURACIÓN.

La configuración y monitoreo de *HA OSCAR* se lo va realizar con la herramienta *webmin*, la cual se instala por defecto con el paquete *HA OSCAR 1.2*, esta herramienta presenta diversas opciones de configuración y monitoreo, ver Figura 3-17.

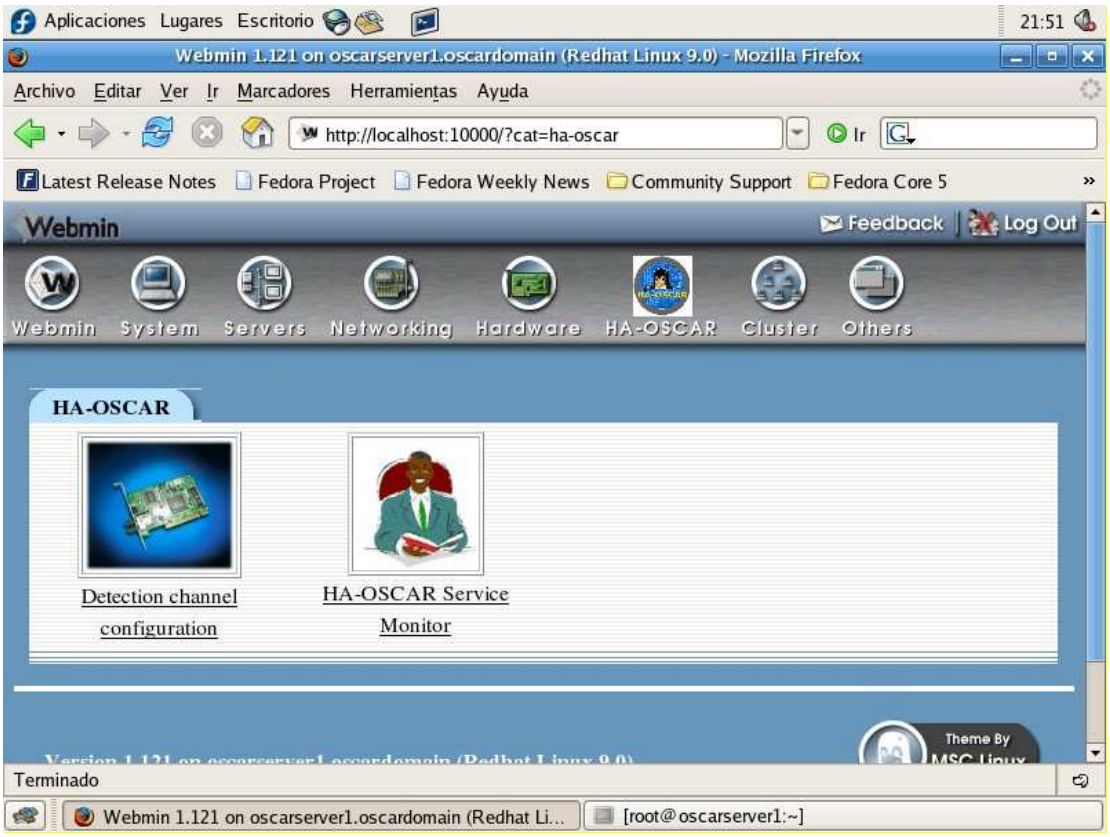


Figura 3-17: HA-OSCAR

De estas dos opciones se van analizar las herramientas de configuración más importantes.

3.3.4.1 Detección del canal de configuración

Presenta cuatro opciones de configuración, las cuales deben realizarse en los servidores primario y *standbye.*

- Añadir o editar interfaces de red para *HA-OSCAR.*
- Configuración del canal en el servidor primario.
- Configuración del canal en el servidor *standbye*.
- Nombre de Host.

Figura 3-18: Detección del canal de configuración HA-OSCAR

3.3.4.1.1 Añadir o editar interfaces de red para HA-OSCAR.

Por defecto al activar esta opción se presentan interfaces activas como son *eth0, eth1* y *loopback,* este proceso se lo debe realizar en los dos servidores, el primario y el *standbye*. En este paso es necesario configurar interfaces con direcciones IP virtuales para las redes privadas y públicas, para ello se debe añadirlas, la Figura 3-19 muestra lo comentado:

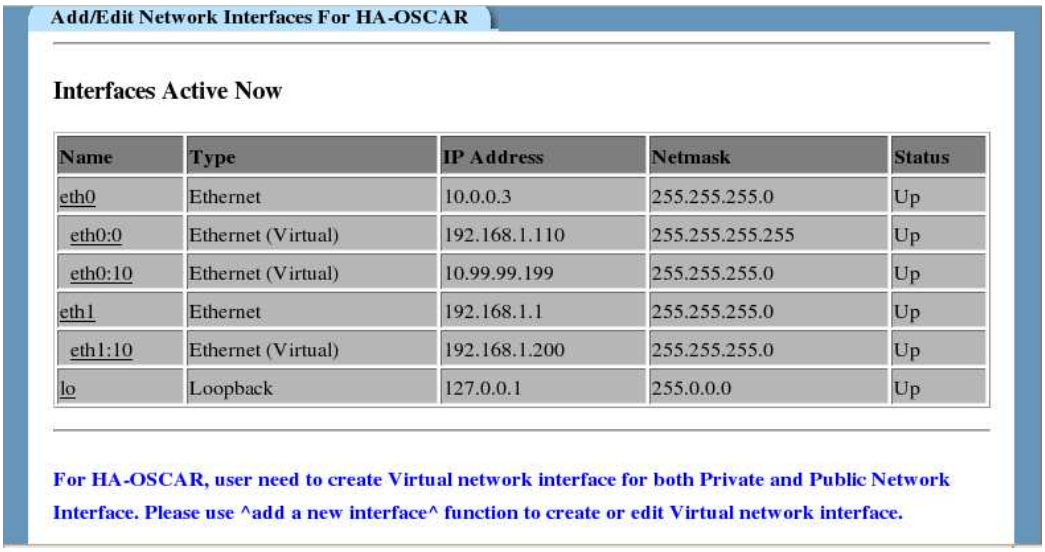


Figura 3-19:Añadir o editar interfaces de red para HA-OSCAR

Posteriormente se debe añadir la interfaz virtual local y pública, tal como se muestra en la Figura 3-20.

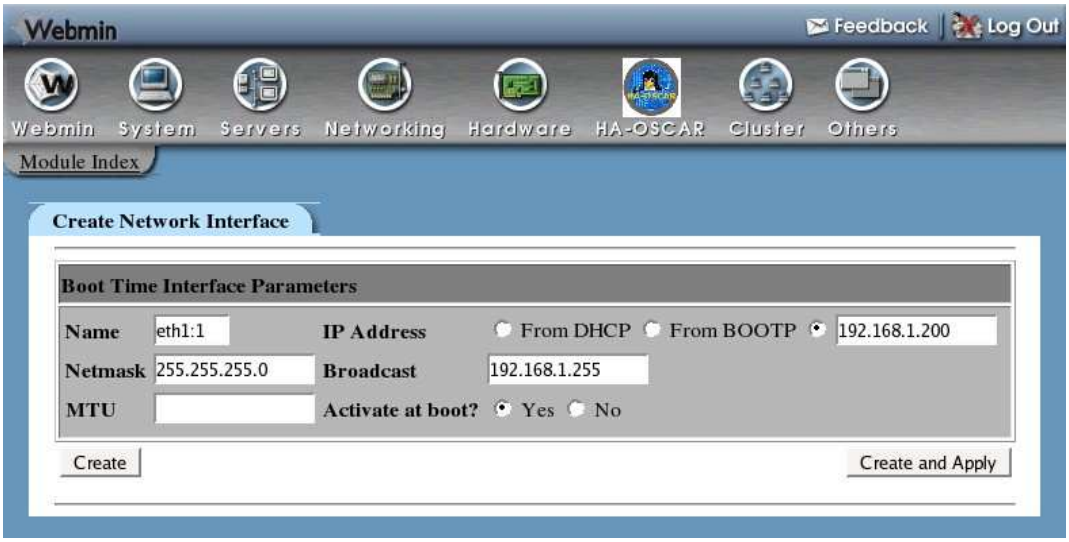


Figura 3-20: Creación de interfaces de red

3.3.4.1.2 Configuración del canal en el servidor primario

Después de crear las interfaces virtuales de red, es necesario definir la interfaz de detección de canal para *HA-OSCAR*. La Figura 3-21 muestra el mecanismo a seguir:

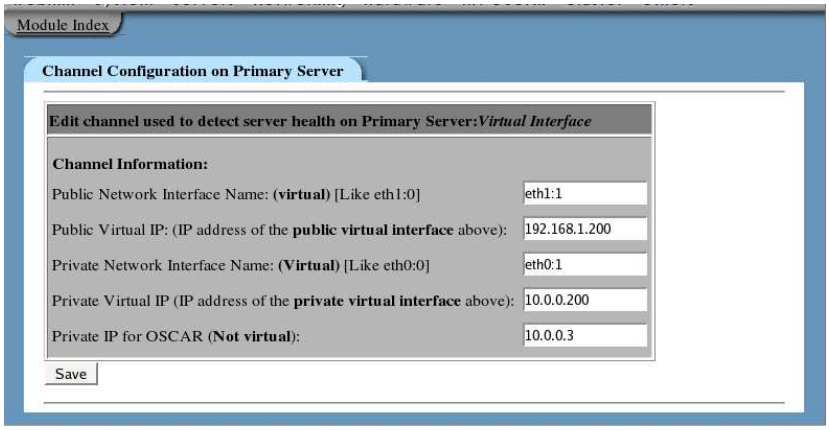


Figura 3-21: Configuración del canal en el servidor primario

3.3.4.1.3 Configuración del canal en el servidor standbye.

De igual manera se debe realizar la configuración en el servidor de *standbye*, el proceso es similar al visto anteriormente, solo que se debe añadir otros parámetros de configuración que se muestran en la Figura 3-22.

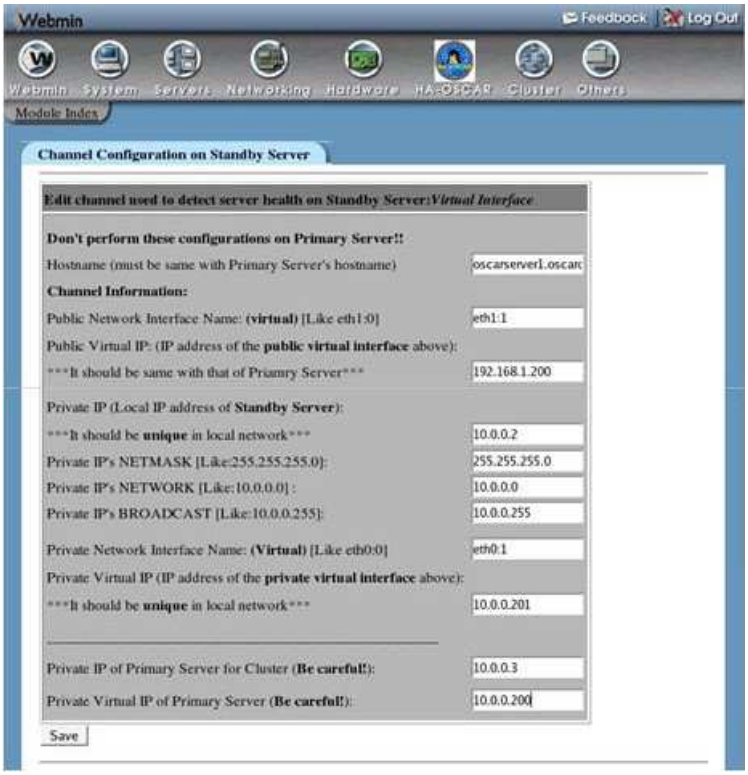


Figura 3-22: Configuración del canal de Standby

3.3.4.1.4 Nombre de Host

Esta opción muestra las direcciones IP de los *host* configurados y sus respectivos nombres, la Figura 3-23 muestra lo dicho:

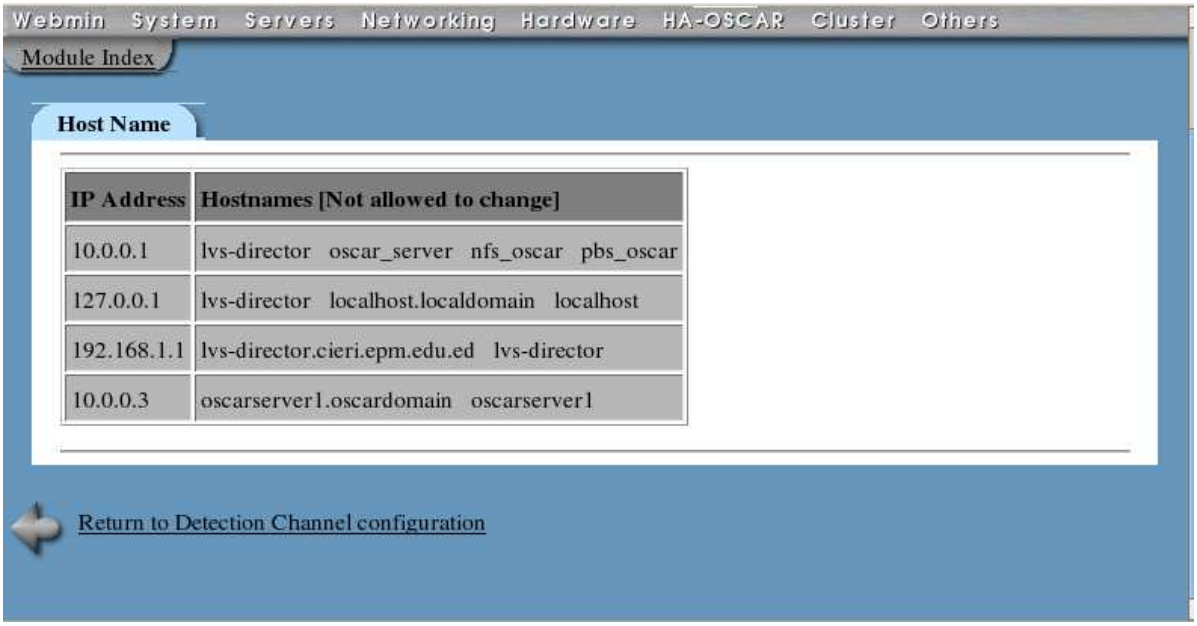


Figura 3-23: Nombre del *Host*

3.3.4.2 Servicio de monitoreo de *HA-OSCAR*

La herramienta *webmin* instalada automáticamente con el paquete *HA- OSCAR* provee varios mecanismos de monitoreo, de los cuales se va a revisar la utilidad de los más importantes. La Figura 3-24 muestra las diferentes opciones de monitoreo existentes.

La Figura 3-25 muestra el recuadro de configuración global, donde se debe especificar el tipo de autenticación de usuario, el directorio de alertas y monitoreo de programas, los cuales vienen ya especificados por defecto por *HA-OSCAR*.

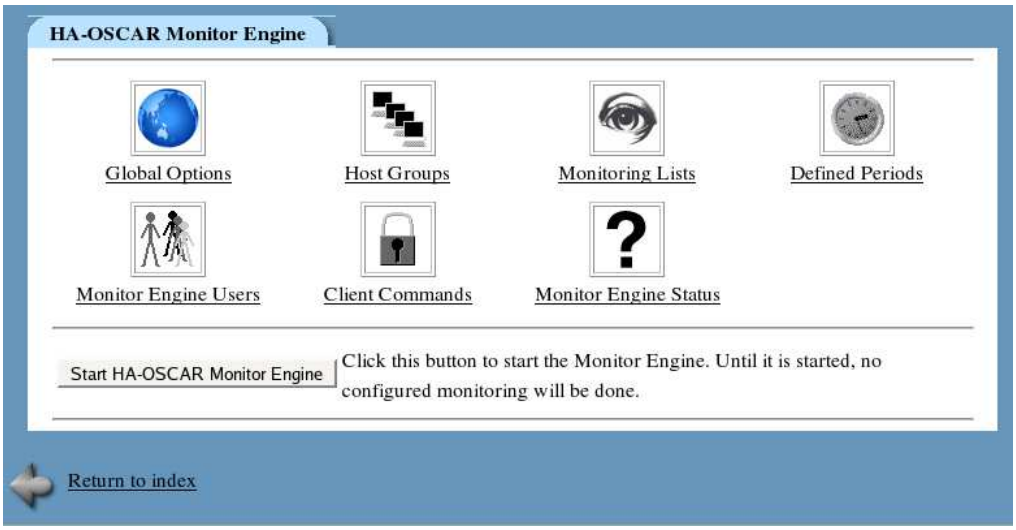


Figura 3-24: Servicio de Monitoreo de HA OSCAR

Estos programas de alertas y monitoreo son los que van a permitir tener un adecuado control de todo es sistema, es cuestión de programarlos de acuerdo a las necesidades que el administrador necesite monitorear.

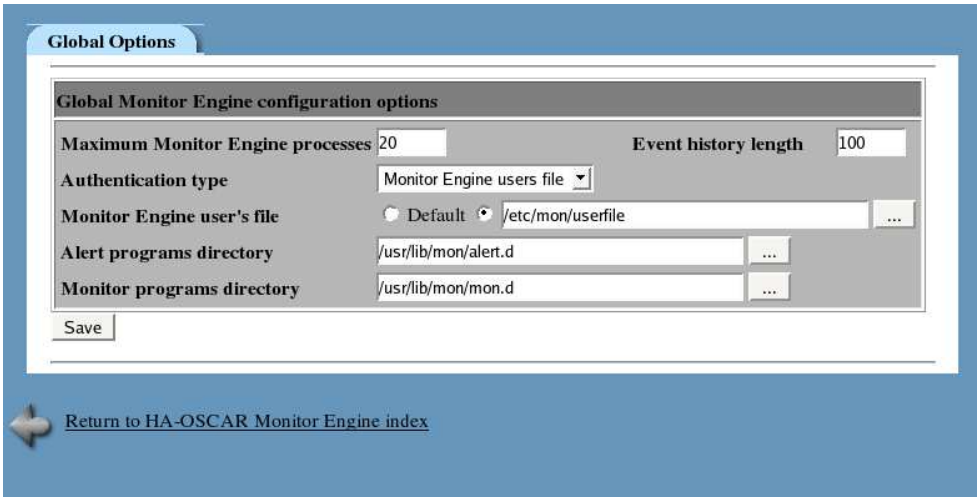


Figura 3-25: Opciones De configuración Global

En la Figura 3-26 se debe especificar la dirección IP del servidor primario y del servidor *standbye,* este grupo de host es importante determinarlo para efectos de monitoreo de funcionamiento de éstos.

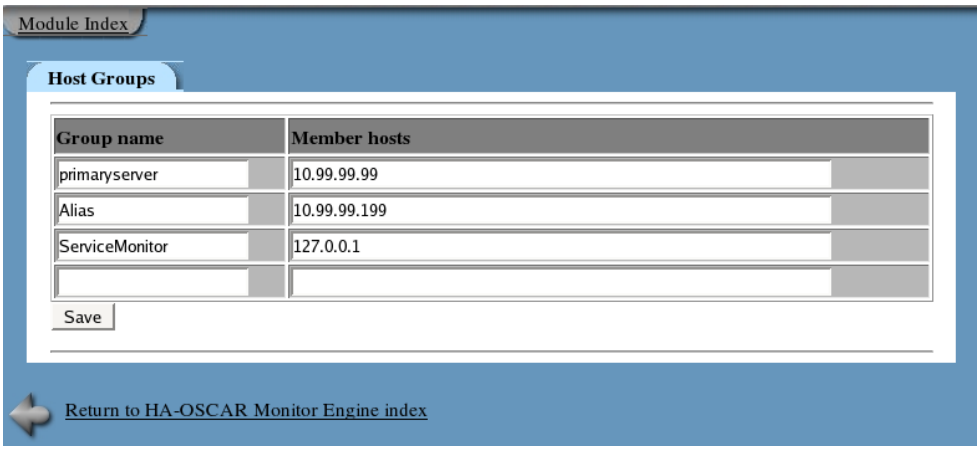


Figura 3-26: Grupos de Host

Es importante definir una lista de monitoreo, tanto para el servidor primario, como para el *standbye (alias),* en la Figura 3-27 muestra un ejemplo en el que al servidor primario se va realizar un *ping* cada 2 segundos, de igual manera al servidor alias se va realizar cada 3 segundos, además se puede añadir otros servicios los cuales se quiere monitorear, esto presionando *add service.*

Figura 3-27: Grupos de Monitoreo

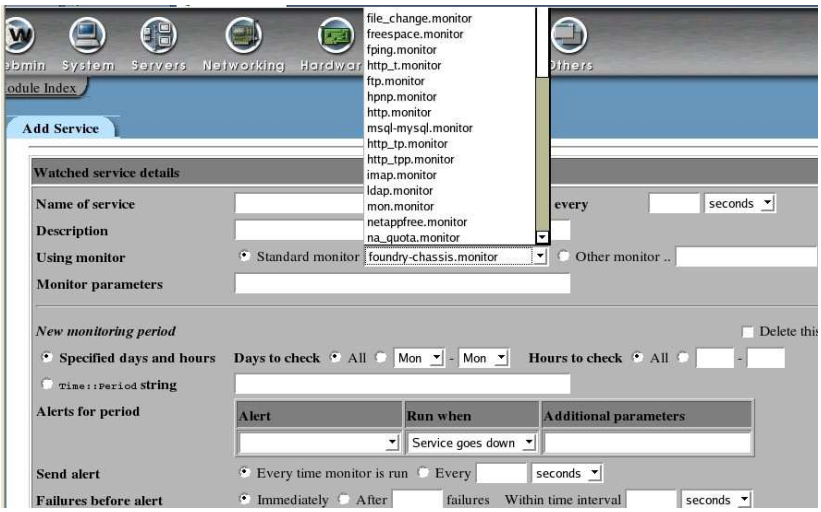


Figura 3-28: Selección de servicios a Monitorear

La Figura 3-28 muestra otros servicios que se pueden añadir para poderlos monitorear, se debe especificar el nombre. Descripción, periodo en segundos/minuto/horas en que se va realizar el monitoreo, el tipo de monitoreo, así como se puede calendarizar periodos de tiempo en la semana, días u horas en que se desea que se efectúe el monitoreo especificado.

La herramienta de monitoreo brinda gran ayuda para programar periodos de monitoreo generales (a todo es sistema según lo especificado para monitorear tanto a nodo principal como a su alias), pudiendo hacerlo todos los días ó ciertos días de la semana, a toda hora ó durante un periodo especificado. La Figura 3-29 muestra el recuadro de configuración.

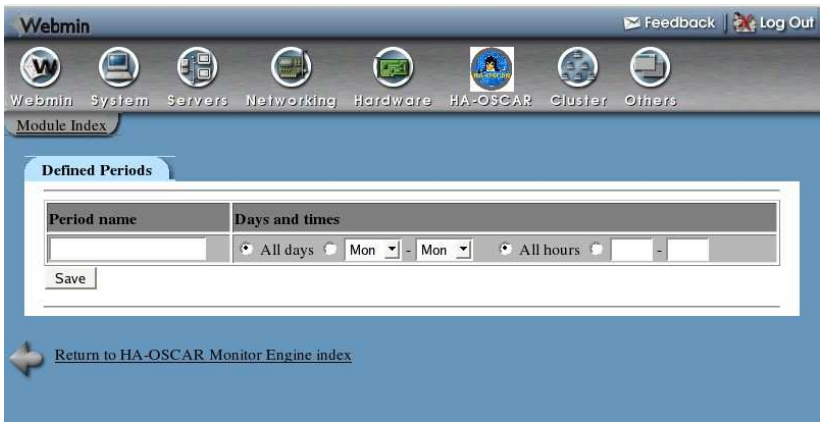


Figura 3-29: Definición de Periodos de Monitoreo

3.4 DISEÑO DE APLICACIÓN WEB E IMPLEMENTACIÓN DE SERVICIOS (RESOLUCIÓN DE NOMBRES).

3.4.1 DISEÑO DE APLICACIÓN WEB

La aplicación Web a realizarse es aquella que va permitir enviar correos electrónicos utilizando los servicios de correo que ofrece el *cluster* . La aplicación es desarrollada en lenguaje de programación java, utilizando *Sun Java creador 2 Update 1*[55] creado por la empresa *Sun Microsystem.*

La figura muestra la interfaz de programación de *Sun Java Creador* donde se desarrolló la aplicación.

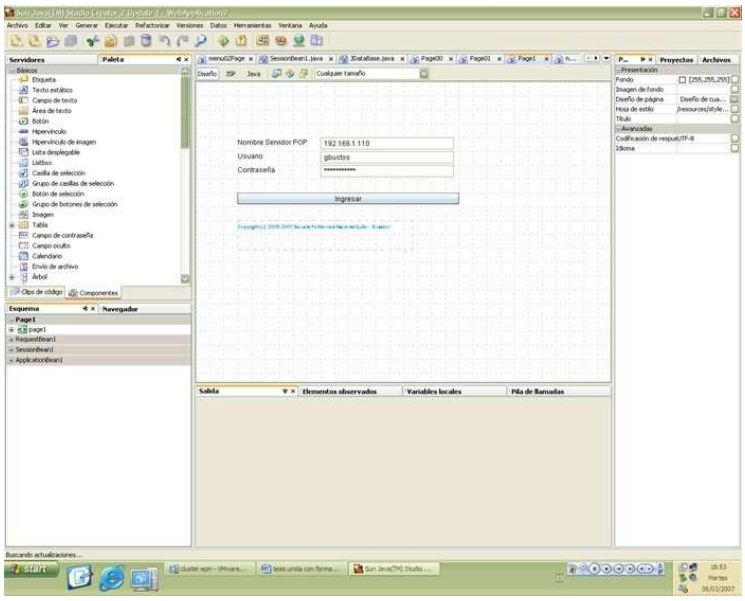


Figura 3-30: Interfaz Sun Java Creator

Una vez terminada la aplicación se procede a compilarla, la figura mostrada indica la interfaz principal de la aplicación donde el usuario debe especificar la dirección IP del servidor de correo, así como el usuario y su contraseña.

[55] Ver sitio Web: http://es.sun.com/

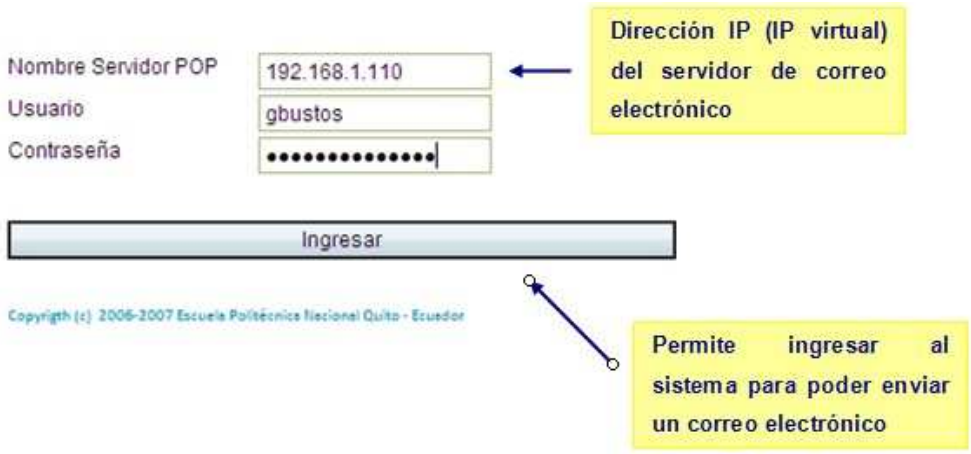


Figura 3-31: Pantalla Principal de la Aplicación Web

Una vez ingresado el usuario al sistema, se presenta la pantalla donde debe ingresar el destinatario y el contenido del correo a enviarse.

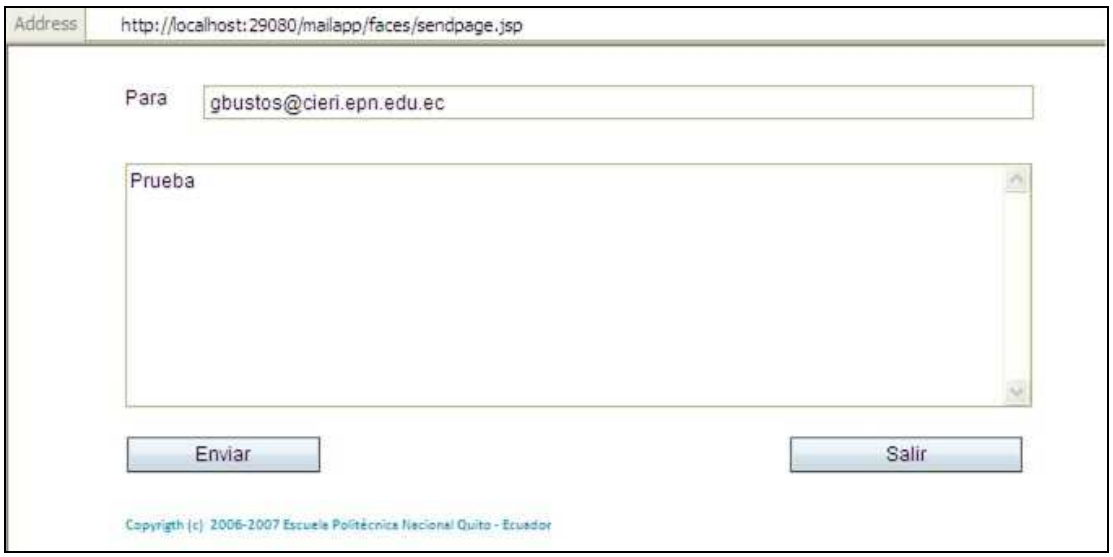


Figura 3-32: Interfaz web para envío de correo electrónico

Una vez ingresados los datos de correo electrónico se debe enviarlo, este se envía a la interfaz virtual del *cluster* y el nodo director lo envía al servidor real disponible para que sea éste quien lo procese.

3.4.2 SERVICIO DE CORREO ELECTRÓNICO *POSTFIX* [6]

3.4.2.1 Introducción

Postfix es un agente de transporte de correo electrónico MTA (*Mail Transport Agent*), herramienta muy útil para configurar el servicio de correo electrónico, el cual pretende ser rápido, fácil de administrar y seguro, a la vez que suficientemente compatible con otras herramientas de correo electrónico como Sendmail [56].

Postfix es un servicio de correo electrónico que consta de dos partes bien diferenciadas: la primera se trata del usuario, y la segunda aquella que se encarga de transportar los mensajes del origen al destino. Existen varios componentes que tienen una función importante durante el proceso de funcionamiento del servicio de correo electrónico, estos son:

- MUA (*Mail User Agent*): Programa que permite leer y escribir correos. Existen varias presentaciones de MUAs existentes, pero las más importantes se detallan a continuación: en ambientes Linux *mail*, *kmail* (entorno KDE), *thunderbird*; entre otras. En ambientes *Windows Netscape, Messenger, Microsoft Outlook Express;* entre otras.

- MTA (Mail Transport Agent): Programa encargado de recoger mensajes y enviarlos, el cual es el encargado de esperar peticiones de MTAs y MUAs para atenderlas. En Linux MTA se presenta como uno o más demonios encargados de monitorear constantemente y servir a dichas peticiones. El MTA más famoso y utilizado es *sendmail*, pero existen otros MTAs como *Postfix, QMail*[57]; entre otros.

El proceso de comunicación de estos componentes se detallan en la Figura 3-33:

---

[56] Visitar el sitio Web: http://www.sendmail.org/

[57] Visitar el sitio Web: http://es.tldp.org/Tutoriales/GUIA_QMAIL/html/

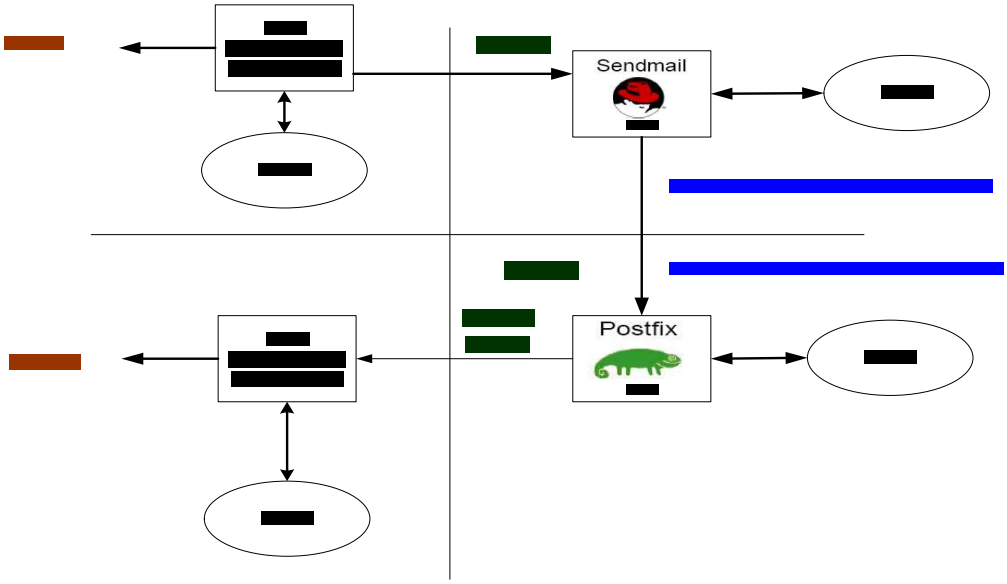


Figura 3-33 Funcionamiento de *Postfix*

Como se puede visualizar existen otros componentes fundamentales en el funcionamiento de *postfix* como son SMTP, POP, IMAP, pop3d, imad. A continuación se da una breve explicación de éstos:

- STMP (*Simple Mail Transfer Protocol*). Protocolo simple de transferencia de correo electrónico. Protocolo de red utilizado para el intercambio de mensajes de correo electrónico entre computadoras o distintos dispositivos de comunicación.

- POP (*Post Office Protocol)*. A diferencia del protocolo SMTP, POP no necesita una conexión permanente a Internet, puesto que es en el momento de la conexión cuando solicita al servidor el envío de la correspondencia almacenada en el servidor para dicho usuario. La situación actual es que se utiliza el protocolo SMTP para el envío de correo y para la recepción de correo se utiliza el protocolo POP, pero ya en su tercera versión desde su aparición, el POP3.

- POP3 (*Post Office Protocol* version 3). Es un protocolo estándar para recuperar correo electrónico. El protocolo POP3 controla la conexión entre

un cliente de correo electrónico POP3 y un servidor donde se almacena el correo electrónico.

- IMAP *(Internet Message Access Protocol*). Protocolo de acceso a mensajes de Internet, es otro método que utilizan las aplicaciones cliente de correo electrónico para obtener acceso a los mensajes almacenados remotamente. La versión más reciente es la IMAP4.

3.4.2.2 Configuración del Servicio de Correo Electrónico *Postfix*.

Para el proceso de configuración del servicio de correo electrónico *Postfix* se debe previamente verificar si el paquete de *Postfix* se encuentra instalado en el sistema operativo, en caso de no estarlo se debe utilizar la versión correspondiente para *Fedora Core 5*, con las siguientes características:

- Versión : 2.2.10
- fedora-release-5-5 (fedora-5.0)
- Release : 4.fc5
- Fuente RPM: postfix-2.2.10-4.fc5.src.rpm

Una vez instalado el paquete *postfix* se procede a configurar mencionado servicio:

Configurando los registros DNS MX.

1. En el servidor master, añadir el registro MX al archivo de nombre de zona y reiniciar el servicio *named.*

2. Para poder comprobar que los cambios tuvieron efecto, usar el comando *dig.*

3. Configurar *postfix* para aceptar correos para este dominio, desde todos los sistemas en la red. Cambiar las siguientes líneas:

myhostname = lvs-realserver00.cieri.epn.edu.ec cieri.epn.edu.ec
mydomain = cieri.epn.edu.ec
myorigin = $mydomain
inet_interfaces = all
mydestination = $myhostname, localhost.$mydomain, $mydomain

4. Iniciar el servicio *Postfix.*

Una vez activado y configurado el servicio de *postfix* se debe habilitar el servicio pop3 el cual va ser útil para la descarga del correo de cada cliente.

5. Habilitar y luego restaurar el demonio *dovecot* para pop3.
6. Se debe chequear si el demonio de pop3 esta escuchando en el puerto 110.

Una vez habilitado el protocolo POP3, es fundamental tomar en cuenta que el *password* de POP3 se envía en texto plano, es decir sin ninguna protección. Para ello es fundamental agregarle seguridades mediante el uso de encripción SSL (*Secure Socket Layer*). Para ello se debe utilizar el protocolo POP3 seguro POP3s.

7. Habilitando y probando POP3s. Es importante cerciorarse que el certificado firmado por *dovecot* existe en la localización apropiada. Este certificado fue generado cuando *dovecot* fue instalado. Luego de ello se debe habilitar el demonio *dovecot* para POP3S.

8. Posteriormente se debe verificar si el demonio de POP3S está escuchando en el puerto 995.

   Se debería verificar que el demonio *dovecot* de *Fedora* es el que escucha en el puerto 995.

La configuración real del servicio de correo electrónico se encuentra detallada en el anexo C.

3.4.3 SERVICIOS DE RESOLUCIÓN DE NOMBRES. [7]

3.4.3.1 Introducción.

DNS es una abreviatura para sistema de nombres de dominio (Domain Name System), sistema utilizado para asignar nombres a equipos y servicios de red que se organiza en una jerarquía de dominios. La asignación de nombres DNS se utiliza en las redes TCP/IP para localizar equipos y servicios con nombres descriptivos. Cuando un usuario escriba un nombre DNS en una aplicación, los servicios DNS podrán traducir el nombre a otra información asociada con el mismo, como una dirección IP.

Por ejemplo, la mayoría de los usuarios prefieren un nombre descriptivo, fácil de utilizar, como ejemplo.microsoft.com para localizar un equipo (como un servidor Web o de correo electrónico) en la red. Un nombre descriptivo resulta más fácil de aprender y recordar. Sin embargo, los equipos se comunican a través de una red mediante direcciones numéricas. Para facilitar el uso de los recursos de red, los sistemas de nombres como DNS proporcionan una forma de asignar estos nombres descriptivos de los equipos o servicios a sus direcciones numéricas.

La siguiente ilustración muestra un uso básico de DNS, consistente en la búsqueda de la dirección IP de un equipo basada en su nombre.

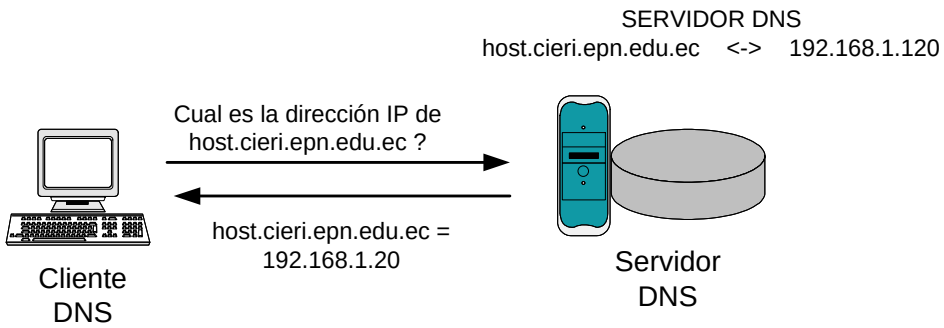


Figura 3-34: Uso básico DNS

En este ejemplo, un equipo cliente consulta a un servidor DNS, preguntando la dirección IP de un equipo configurado para utilizar host.cieri.epn.edu.ec como nombre de dominio. Como el servidor puede utilizar la base de datos local para responder la consulta, contesta con una respuesta que contiene la información solicitada, un registro de recursos de host (A) contiene la información de dirección IP para host.cieri.epn.edu.ec. El registro de tipo A, es aquel que relaciona un nombre totalmente cualificado con una dirección IP.

El nombre de dominio principal, puede ser dividido en zonas, una zona se inicia como una base de datos de almacenamiento para un único nombre de dominio DNS. Si se agregan otros dominios bajo el dominio utilizado para crear la zona, estos dominios pueden formar parte de la misma zona o pertenecer a otra zona.

Existen dos tipos de zonas, la directa y la inversa, la directa es utilizada para traducir nombres de hosts a direcciones IP, mientras la inversa es utilizada para traducir IPs en nombres de dominio.

Dentro de la configuración de estas zonas existen diferentes registros utilizados en servicios de resolución de nombres DNS, los cuales se detallan a continuación:

- A = Address – (Dirección) Este registro se usa para traducir nombres de hosts a direcciones IP.

- CNAME = Canonical Name – (Nombre Canónico) Se usa para crear nombres de hosts adicionales, o alias, para los hosts de un dominio.

- NS = Name Server – (Servidor de Nombres) Define la asociación que existe entre un nombre de dominio y los servidores de nombres que almacenan la información de dicho dominio. Cada dominio se puede asociar a una cantidad cualquiera de servidores de nombres.

- MX = Mail Exchange – (Intercambiador de Correo) Define el lugar donde se aloja el correo que recibe el dominio.

- PTR = Pointer – (Indicador) También conocido como 'registro inverso', funciona a la inversa del registro A, traduciendo IPs en nombres de dominio.

3.4.3.2 Configuración del Servicio de Resolución de Nombres DNS.

Para la configuración del servicio de resolución de nombres DNS, primeramente se escoge el nombre de dominio a utilizarse, para el presente caso se utilizó el dominio "**cieri.epn.edu"**. Luego de esto se debe realizar los siguientes pasos para tener configurado el servicio:

1. En primer lugar se debe ingresar al sistema como administrador ***root.***

2. Se debe modificar el archivo ***named.conf*.** En el sistema operativo *Fedora* se debe chequear si el paquete ***bind-chroot.RPM*** ha sido instalado, es importante exista, ya que el archivo ***named.conf*** se ejecuta sobre este ambiente, es decir en el directorio **/var/named/chroot/etc/named.conf.**

3. Crear el archivo con el nombre de la zona. Este debería incluir todos los host existentes en el dominio.

4. Crear el archivo de zona IP, de manera similar a como se lo hizo con el archivo del nombre de zona.

5. Posteriormente se debe verificar los archivos de zona local. Estos archivos deben ser instalados de manera automática el momento de instalar el sistema operativo con el servicio DNS habilitado. Estos archivos se encuentran en los siguientes directorios:

6. Copiar el archivo */etc/host* pata crear un *backup*. Luego de ello editar el archivo */etc/host,* eliminando todos los hosts excepto el *localhost* del sistema.

7. Se debe cambiar el nombre del host por el nombre de dominio adecuado.

8. Crear el archivo /etc/resolv.conf

9. Se debe reiniciar el demonio *named* y verificar que se inicie sin problemas.

10. Verificar si los archivos de *backup* fueron realmente creados.

Los pasos que se detallan a continuación son utilizados para configurar un cliente que pertenece al dominio de nombres, se deben realizar únicamente en los clientes más no en los servidores de nombre.

11. Copiar el archivo */etc/hosts* para crear un *backup*. Luego editar este archivo comentando todas las líneas, excepto las relacionadas al *localhost*. Este paso no es necesario, pero se debe asegurar que se esta utilizando el servidor de nombres exclusivamente.

12. Se debe cambiar el nombre del host por el nombre de dominio adecuado.

13. Crear el archivo /etc/resolv.conf

La configuración real del servicio de resolución de nombres DNS se encuentra detallada en el anexo B.

3.5 PRUEBAS CON *BENCHMARK* ECPERF.

ECperf es un generador de tráfico creado con lenguaje de programación java se esta utilizando la versión *ECperf* 1.1[58], posee varias clases las cuales deben ser compiladas con cualquier compilador de java, en este caso se utilizó *java EE SDK* versión 5.

[58] Visitar sitio Web: http://java.sun.com/developer/releases/j2ee/ecperf/

Las pruebas realizadas se detallan a continuación, donde la Figura 3-35 indica la manera como se las realizó. Como se puede visualizar existen los dos servidores reales y los dos nodos directores, todos ellos utilizando el mecanismo de balanceo de carga basado en enrutamiento directo. El cliente es el encargado de generar los requerimientos en los diferentes servicios (DNS, SMTP, POP3, WWW) y mediante el *benchmark* ECperf simular alto tráfico hacia los servicios ofrecidos por el *cluster* en general.

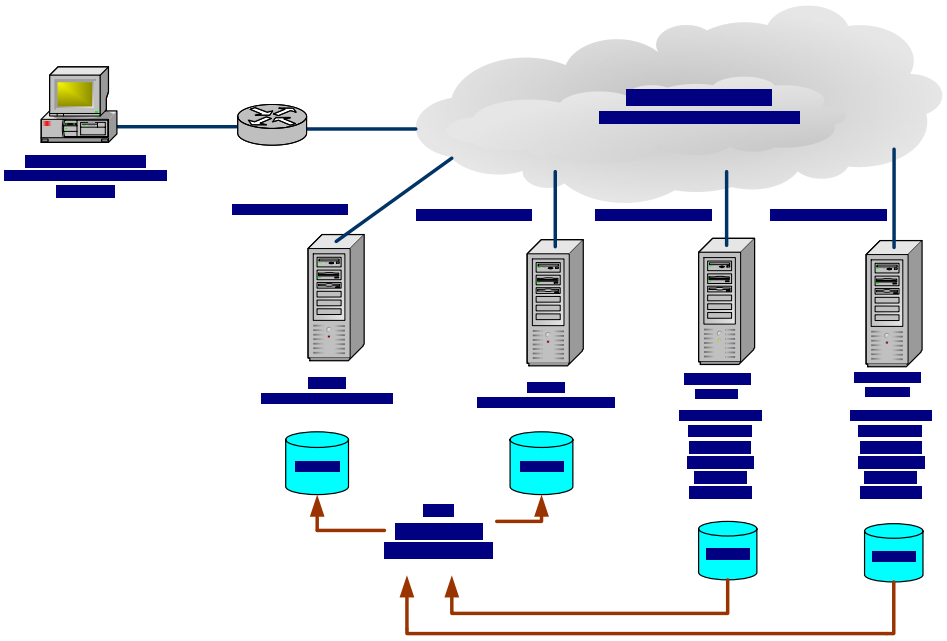


Figura 3-35: Arquitectura de pruebas realizadas

Una vez instalado el generador de tráfico se lo puede manipular mediante una interfaz Web, sobre la cual se especifican los parámetros de configuración para proceder a generar el tráfico necesitado y con ello poder comprobar el funcionamiento adecuado del *cluster.* La Figura 3-36 muestra la interfaz Web principal del generador.

La métrica de rendimiento provista por el benchmark ECPerf es la velocidad efectiva del servidor de aplicaciones (*throughput*) medidas en las interacciones de un Servicio Web por Segundo (SIPS).

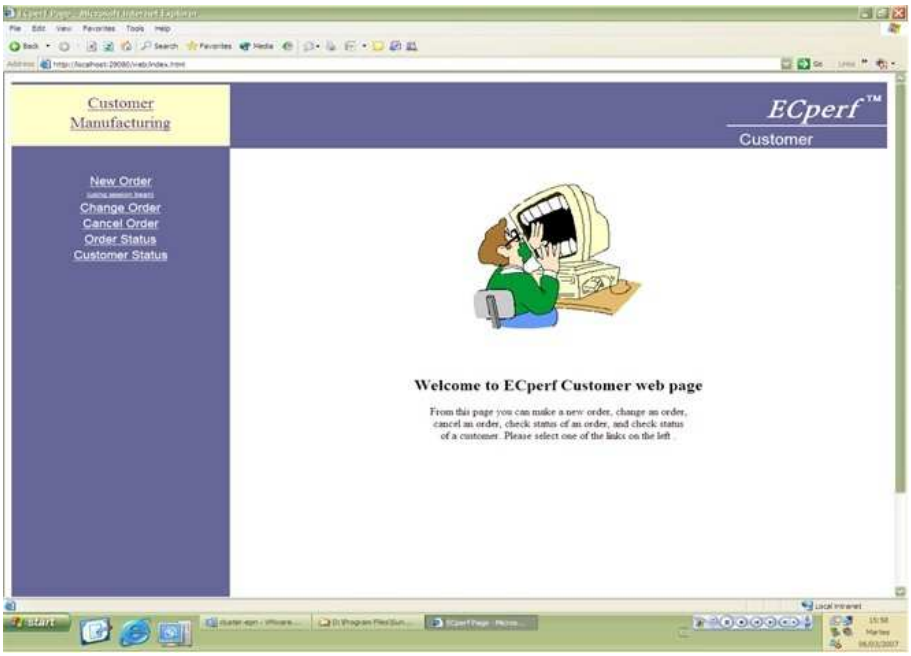


Figura 3-36. Aplicación Web de *bechmark* ECPerf.

Los resultados del bechmark incluyen los SIPS del sistema (servidor de aplicaciones) *cluster*. La tabla 3-7 muestra los resultados obtenidos donde se comparan el número de transacciones emuladas respecto al tiempo de respuesta promedio.

Nuevo consumidor	Nuevos productos	Cambio de ítem	Crear orden	
Tiempo de respuesta promedio (Segundos)	Tiempo de respuesta promedio (Segundos)	Tiempo de respuesta promedio (Segundos)	Tiempo de respuesta promedio (Segundos)	Número de Transacciones Emuladas
0,67	0,665	0,67165	0,92	4
0,72	0,698	0,70498	1,89	10
0,83	0,788	0,79588	2,46	13
0,92	0,899	0,90799	3,78	20
0,98	0,991	1,00091	4,00	25
0,89	0,906	0,91506	5,08	30
1,91	2,02	2,0402	7,98	50
2,25	2,21	2,2321	10,02	70
3,88	3,67	3,7067	11,88	100

Tabla 3-7. Resultados obtenidos durante la ejecución de *bechmark* ECPerf.

La Figura 3-37 muestra una representación gráfica del número de transacciones emuladas respecto al tiempo promedio de respuesta, donde se puede observar

que conforme incrementan las transacciones el *throughput* de de las aplicaciones se incrementa en el tiempo.

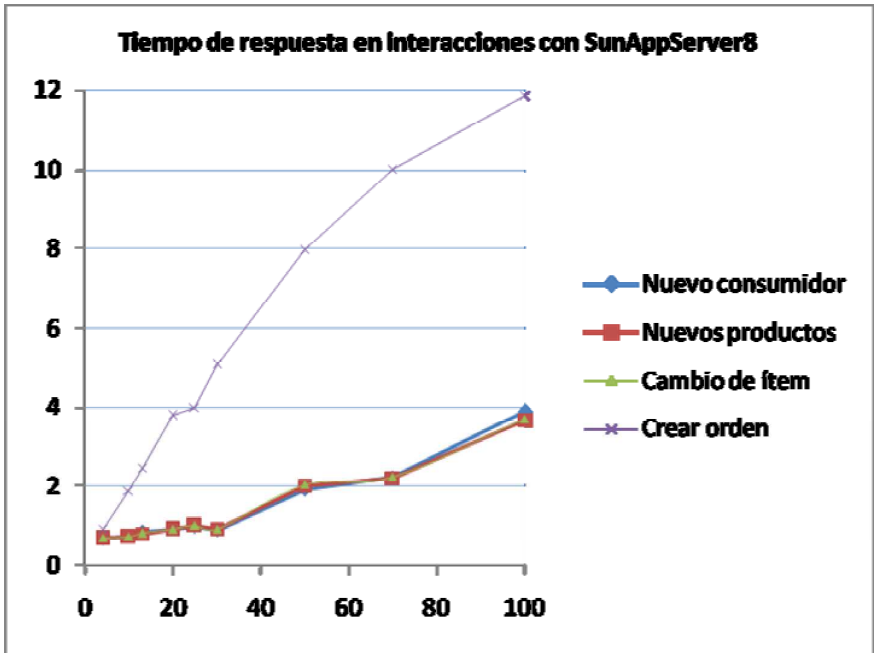


Figura 3-37. Resultados obtenidos durante la ejecución de *benchmark* ECPerf.

3.5.1 PRUEBAS DE FUNCIONAMIENTO DEL *CLUSTER* DE ALTA DISPONIBILIDAD Y BALANCEO DE CARGA.

3.5.1.1 Objetivo.

Como objetivo principal de estas pruebas es la de corroborar el adecuado funcionamiento y configuración del *cluster* de alta disponibilidad y balanceo de carga mediante la utilización de software de código abierto, el cual fue configurado en los servidores reales, así como en el nodo director y *standby.* El mecanismo de pruebas está constituido en dos ambientes, el primero que consiste en corroborar el funcionamiento de LVS para balanceo de carga, y el segundo donde se va utilizar LVS y HA OSCAR para la alta disponibilidad, servicios DNS, Web, correo y el respectivo generador de tráfico.

Para realizar las pruebas de funcionamiento del *cluster* se debe aclarar que se las está realizando con máquinas virtuales (*VMWARE*[59]*)*[5], sobre las cuales se instaló el sistema operativo Linuxm Fedora Core 5. Con esta herramienta se instalaron los nodos directores principal y *standby*, así como los servidores reales. La instalación de estos servidores se lo hizo con las mismas características de hardware y software es decir simulando un *cluster* homogéneo donde los servidores reales poseen la misma configuración y componentes de hardware, de igual manera los servidores principales y *stanby*.

3.5.1.2 Escenario

Para poder corroborar el funcionamiento del *cluster,* se crearon dos escenarios de pruebas los cuales de detallan a continuación:

1. El primer escenario consiste en realizar las pruebas de funcionamiento de LVS, mediante la utilización de la herramienta *telnet* hacia la interfaz virtual del *cluster* y por ende debería responder uno de los servidores reales el cual fue escogido por el nodo director de acuerdo a las técnicas de balanceo de carga utilizadas. La Figura 3-38 muestra la arquitectura de este ambiente de pruebas.

2. La segunda parte de las pruebas se va a probar todo el sistema ya configurado, es decir *LVS* para balanceo de carga, *HA OSCAR* para alta disponibilidad, servicio de resolución de nombres DNS, la aplicación Web creada, correo electrónico, así como las pruebas de funcionamiento ante la existencia de un alto tráfico, esto mediante la utilización del generador de tráfico *ecperf* [60].

---

[59] Ver anexo D

[60] Ver sección 3.5

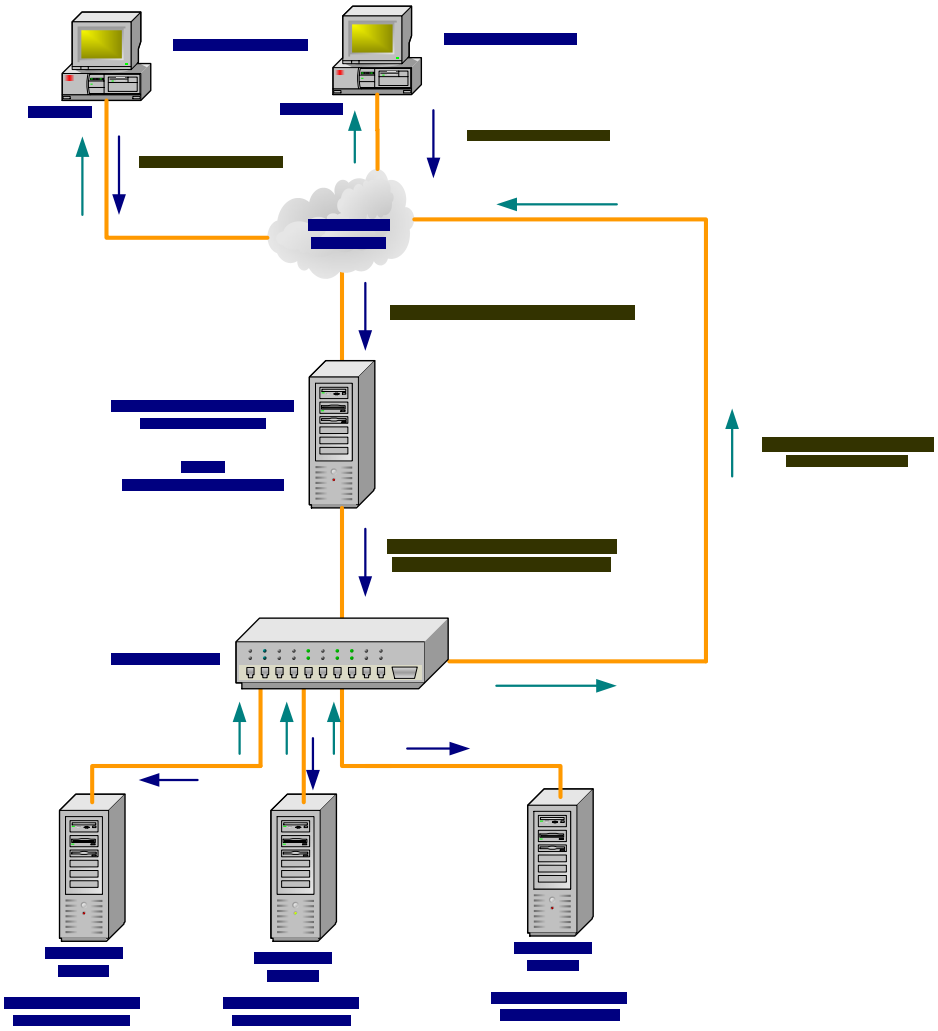


Figura 3-38: Arquitectura de Pruebas con Telnet

La arquitectura del sistema para este segundo ambiente de pruebas se detalla a continuación en la Figura 3-39. Donde se están utilizando 5 PC's de las cuales dos son utilizadas como servidores directores, dos como servidores reales y uno como cliente, además se está utilizando un *switch* marca 3COM mediante el cual se interconectan todos los servidores y cliente.

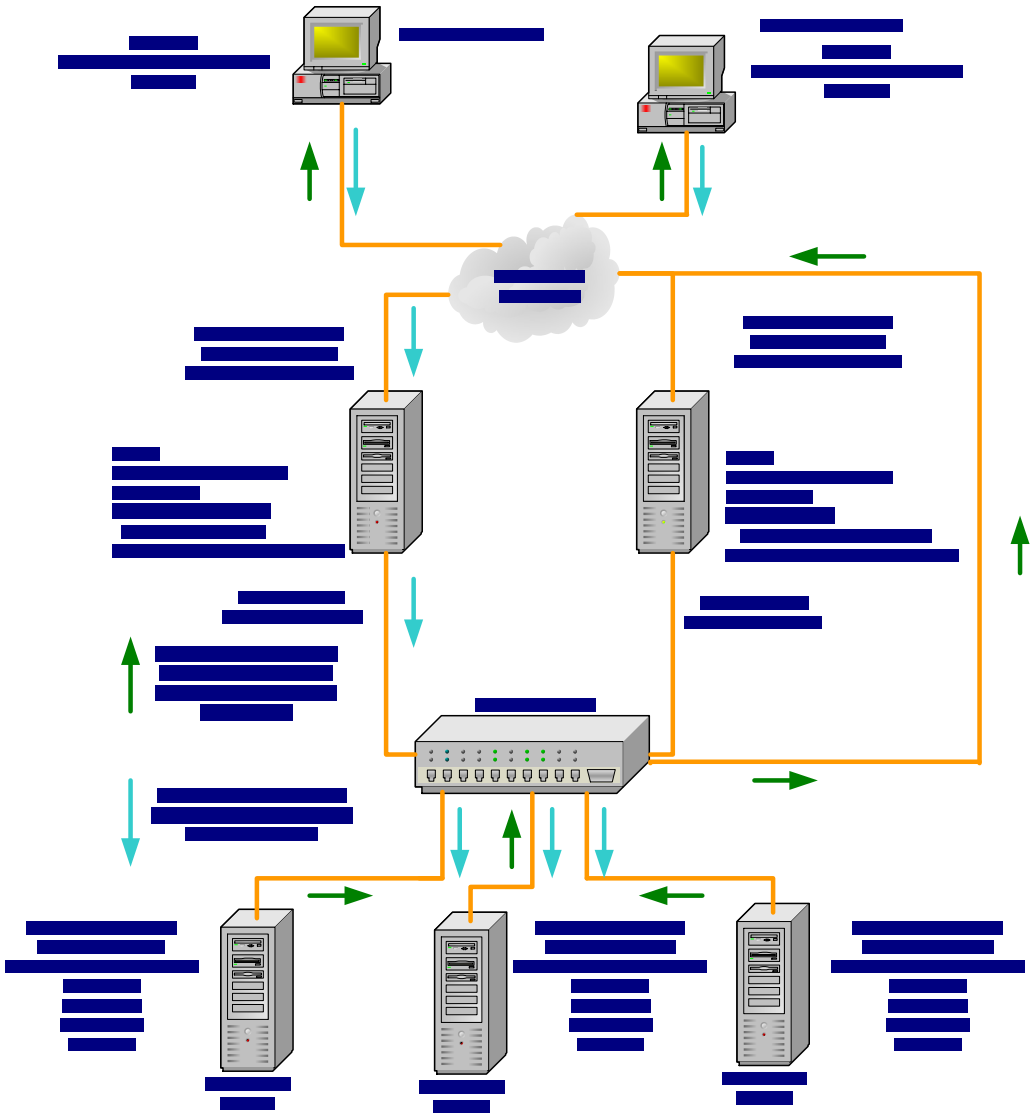


Figura 3-39: Arquitectura de Pruebas generales del Cluster

3.5.1.3 Resultados

3.5.1.3.1 Pruebas de LVS con telnet.

Para la realización de estas pruebas se va utilizar un nodo director el cual va contener la aplicación para configurar los servidores reales utilizando *ipvsadm* mediante una interfaz gráfica creada, se va a utilizar 3 servidores reales de los cuales uno posee deshabilitado requerimientos arp en el *kernel* de su sistema

operativo para deshabilitar requerimientos arp y los otros dos no lo están. También se tienen dos clientes con sistema operativo Windows quienes van a ser los encargados de generar las pruebas mediante *telnet* al la dirección IP virtual. A continuación se presenta la configuración de red de cada uno de ellos:

- Nodo director:

  eth0: 192.168.1.5 interfaz de red real

  eth0:0: 192.168.1.110 interfaz de red virtual

- Servidor real 1:

  eth0: 192.168.1.1 interfaz de red real

  dummy0: 192.168.1.110 interfaz de red virtual

- Servidor real 2:

  eth0: 192.168.1.2 interfaz de red real

  dummy0: 192.168.1.110 interfaz de red virtual

- Servidor real 3: (servidor deshabilitado requerimientos de arp)

  eth0: 192.168.1.3 interfaz de red real

  dummy0: 192.168.1.110 interfaz de red virtual

- cliente 1:

  eth0: 192.168.1.10 interfaz de red real

- cliente 2:

  eth0: 192.168.1.11 interfaz de red real

3.5.1.3.2 Configuración de nodo director

Una vez ejecutada la interfaz de configuración se procede añadir los parámetros requeridos, donde se va tener 3 servidores reales, el puerto de escucha va a ser el 23, y la ponderación de peso de los servidores reales vas ser 1 (valor por defecto) *cluster* homogéneo, la Figura 3-40 muestra lo comentado:

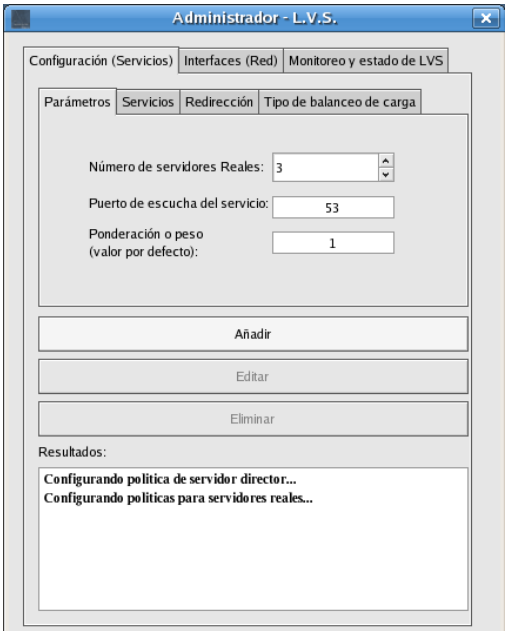


Figura 3-40: Parámetros de Configuración de Servidores reales

Luego de añadir el número de servidores reales, puerto de escucha y peso de cada uno de ellos, se muestran los resultados, donde se especifica que la configuración de los parámetros generales se ha realizado con éxito.

Una vez añadidos los parámetros generales se procede a configurar el tipo se servicio de la aplicación *telnet* el cual es un servicio TCP, la Figura 3-41 muestra lo dicho:

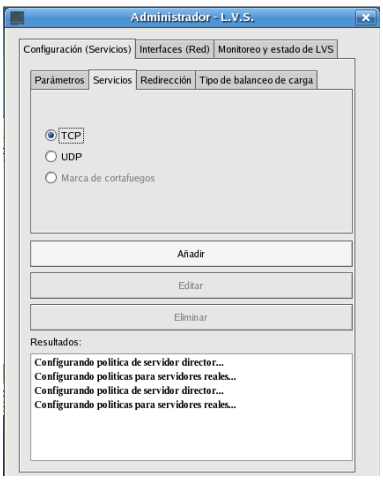


Figura 3-41: Configuración de Servicio

Los resultados mostrados indican que no hubo inconveniente al momento de añadir el tipo de servicio a utilizarse.

El mecanismo de redirección a utilizarse es enrutamiento directo DR, la Figura 3-42 presenta la configuración:

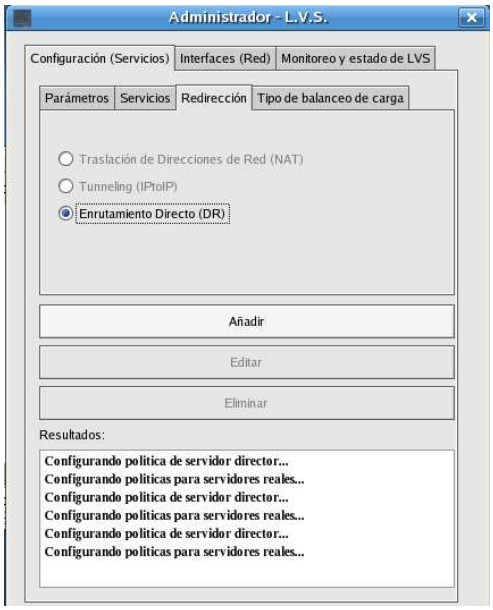


Figura 3-42: Mecanismo de Redirección

El tipo de balanceo de carga a utilizar se muestra en la Figura 3-43, para el caso de las presentes pruebas se esta utilizando *round robin* como tipo de balanceo de carga.

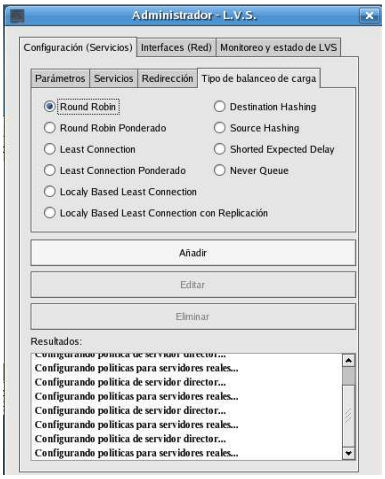


Figura 3-43: Tipo de balanceo de carga

La Figura 3-44 muestra la configuración de la interfaz virtual, la cual debe ser guardada para que ésta se mantenga como una interfaz virtual del sistema.

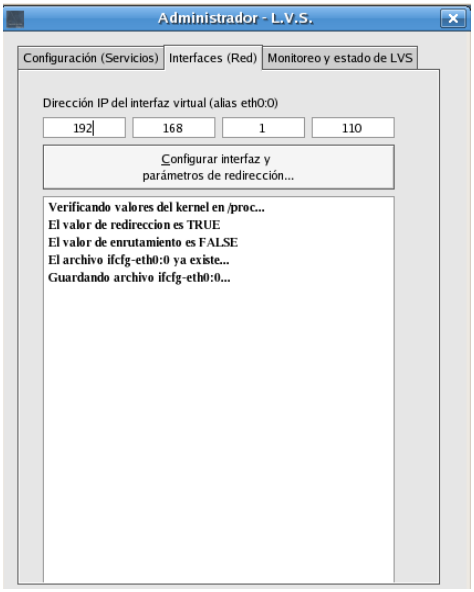


Figura 3-44: Interfaces de Red

Una vez configurado todo el sistema se procede a monitorear el estado del *cluster,* en primer lugar en la Figura 3-45 se muestra las estadísticas del sistema donde se muestra la interfaz virtual y su puerto asociado, así como los servidores reales configurados y su respectivo puerto sobre el cual funcionan.

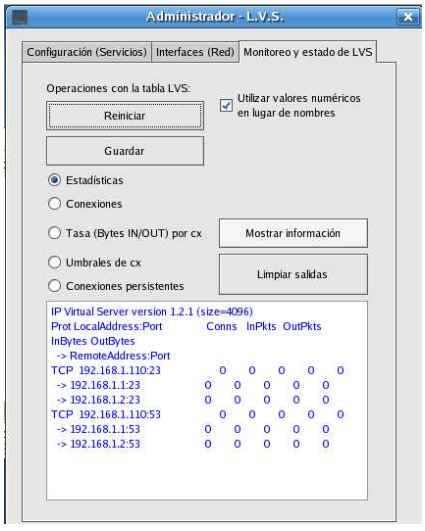


Figura 3-45: Monitoreo y estado de LVS

Además las estadísticas indican las conexiones existentes en cada servidor real, así como los paquetes entrantes y salientes.

3.5.1.3.3 Resultados utilizando telnet

Pruebas con clientes Windows.

Existen dos clientes con sistema operativo Windows, de estos se va a generar un *telnet* a la interfaz virtual del *cluster* y uno de los servidores reales disponibles debe responder a mencionado requerimiento, la Figura 3-46 muestra los requerimientos realizados por los clientes.

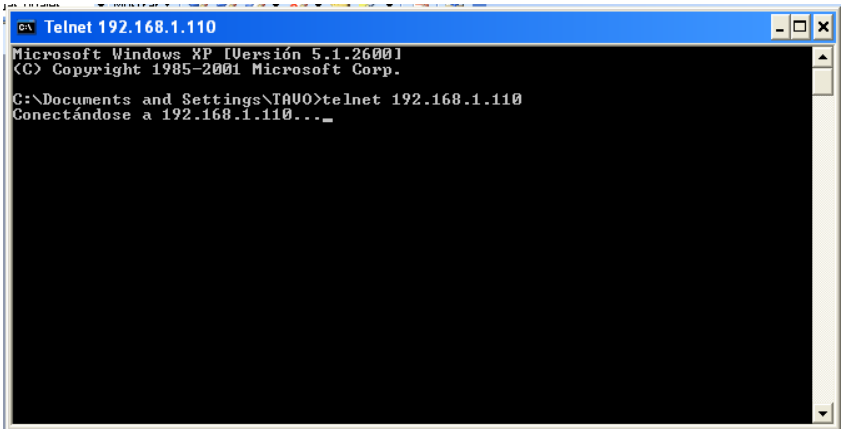


Figura 3-46: Intento de conexión

Una vez generado el telnet la Figura 3-47 muestra que el cliente tuvo éxito en su requerimiento.

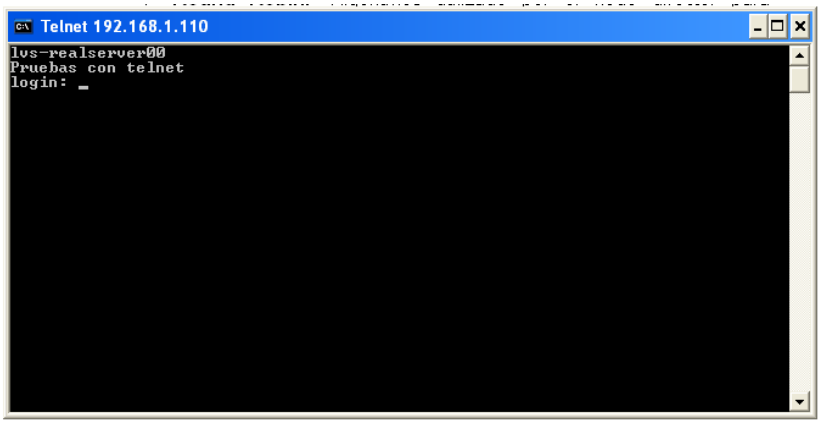


Figura 3-47: Conexión de telnet satisfactoria

La Figura 3-48 muestra las estadísticas de las conexiones existentes, en esta información se puede visualizar el número de conexiones existentes en la interfaz virtual, paquetes de entrada y salida, así como información del número de bytes de entrada y salida generados por los servidores reales presentes en el *cluster*.

Es importante hacer notar que se hicieron 4 intentos de conexión donde se puede observar que el servidor real 1 y 2 tienen una conexión, mientras que el servidor real 3 posee 2 conexiones. Con ello se puede constatar el adecuado funcionamiento del tipo de balanceo utilizado (*round robin*), ya que el nodo director determino que el orden de paso de requerimientos es el adecuado asingnando el siguiente orden :

1. Servidor real 3
2. Servidor real 2
3. Servidor real 1

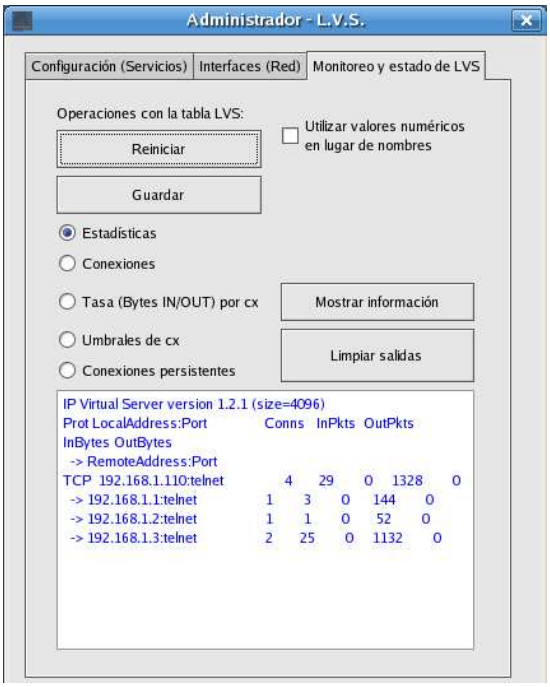


Figura 3-48: Estadísticas de Conexiones existentes con Telnet.

La Figura 3-49 muestra el monitoreo de las conexiones existentes, donde se puede visualizar si la conexión fue estabilizada o no, la dirección IP del servidor real que está haciendo el requerimiento, la dirección IP virtual y la dirección IP del servidor real que esta atendiendo las peticiones.

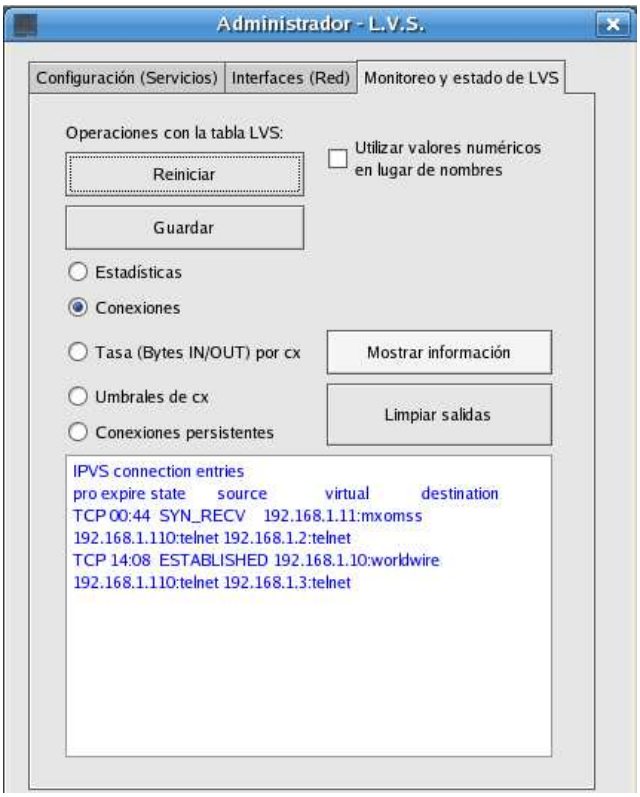


Figura 3-49: Monitoreo de conexiones existentes con Telnet

Es importante hacer notar que uno de los campos de la información mostrada indica si se estableció o no la conexión, para el presente ejemplo indica que el cliente 1 (192.168.1.10) se conectó al servidor real 3 (192.168.1.3) y pudo establecer una conexión es decir pudo generar un telnet a la dirección IP virtual, mientras que el cliente 2 se conectó al servidor real 2 y no pudo establecer la conexión.

Esta prueba permite corroborar que el servidor real 3 no presenta conflictos de arp ya que presenta deshabilitada la opción de generar requerimientos de este tipo (su *kernel parchado)* entre la dirección virtual del nodo director y el servidor

real. En cambio el servidor real 2 el cual no posee deshabilitada la opción de arp presenta conflictos entre las direcciones virtuales configuradas y la presente en el nodo director, razón por la cual el requerimiento no sabe a cual de ellas conectarse.

Una vez terminadas las pruebas se puede concluir que el sistema de balanceo de carga está funcionando adecuadamente, ya que se pudo visualizar en el monitoreador de LVS que todos los servidores reales tienen conexiones presentes, además se pudo constatar que el tipo de balanceo de carga basado en *Round Robin* funciona correctamente ya que las conexiones se fueron asignando de manera ordenada, primero al servidor real 1 luego al 2 y así sucesivamente.

Es importante mencionar que durante las pruebas se tuvieron tiempos de respuesta considerables, ya que no todos los servidores reales tenían deshabilitados requerimientos de arp generando conflictos con la dirección IP virtual del sistema.

3.5.1.4 Pruebas generales del *cluster* de alta disponibilidad y balanceo de carga.

Una vez configurado todo el *cluster* de alta disponibilidad y balanceo de carga, se va a mostrar el funcionamiento adecuado del sistema, así como se procederá a realizar algunas pruebas que permita demostrar el adecuado funcionamiento de éste. A continuación se presenta la configuración de red de cada uno de los servidores utilizados en estas pruebas:

- Nodo director principal:

  eth0: 192.168.1.5 interfaz de red real

  eth0:0: 192.168.1.110 interfaz de red virtual

- Nodo director principal:

  eth0: 192.168.1.6 interfaz de red real

  eth0:0: 192.168.1.110 interfaz de red virtual

- Servidor real 1:

  eth0: 192.168.1.1 interfaz de red real

  dummy0: 192.168.1.110 interfaz de red virtual

- Servidor real 2:

  eth0: 192.168.1.2 interfaz de red real

  dummy0: 192.168.1.110 interfaz de red virtual

- Servidor real 3: (servidor deshabilitado requerimientos de arp)

  eth0: 192.168.1.3 interfaz de red real

  dummy0: 192.168.1.110 interfaz de red virtual

- cliente 1:

  eth0: 192.168.1.10 interfaz de red real

- cliente 2:

  eth0: 192.168.1.11 interfaz de red real

El proceso de pruebas se va a realizar por partes, es decir en primer lugar se va generar tráfico mediante *ECperf,* el cual arroja los siguientes resultados en ambientes diferentes. La tabla mostrada indica los resultados arrojados.

Nuevo consumidor	**Nuevos productos**	**Cambio de ítem**	**Crear orden**	
Tiempo de respuesta promedio (Segundos)	**Tiempo de respuesta promedio (Segundos)**	**Tiempo de respuesta promedio (Segundos)**	**Tiempo de respuesta promedio (Segundos)**	**Número de Transacciones Emuladas**
0,67	0,665	0,67165	0,92	4
0,72	0,698	0,70498	1,89	10
0,83	0,788	0,79588	2,46	13
0,92	0,899	0,90799	3,78	20
0,98	0,991	1,00091	4,00	25
0,89	0,906	0,91506	5,08	30
1,91	2,02	2,0402	7,98	50
2,25	2,21	2,2321	10,02	70
3,88	3,67	3,7067	11,88	100

Tabla 3-8: Resultados ECperf

Una vez configurado el generador de tráfico se va proceder a realizar las pruebas de funcionamiento de cada uno de los servicios que el *cluster* ofrece:

3.5.1.4.1 Pruebas de DNS

Para el proceso de pruebas de DNS se configura el puerto 53 en los servidores reales para que acepten peticiones de este tipo, cabe recalcar que en cada uno se estos servidores se configuró el servicio de resolución de nombres. La muestra de tráfico en los servidores se la obtuvo mediante la realización de un telnet al puerto 53 para confirmar que el servicio estaba operativo, también se pudo corroborar el funcionamiento el momento en que se ejecutaban transacciones de pop3, smtp, www. La Figura 3-50 muestra las conexiones, paquetes entrantes y salientes correspondientes a cada servidor real en el puerto especificado.

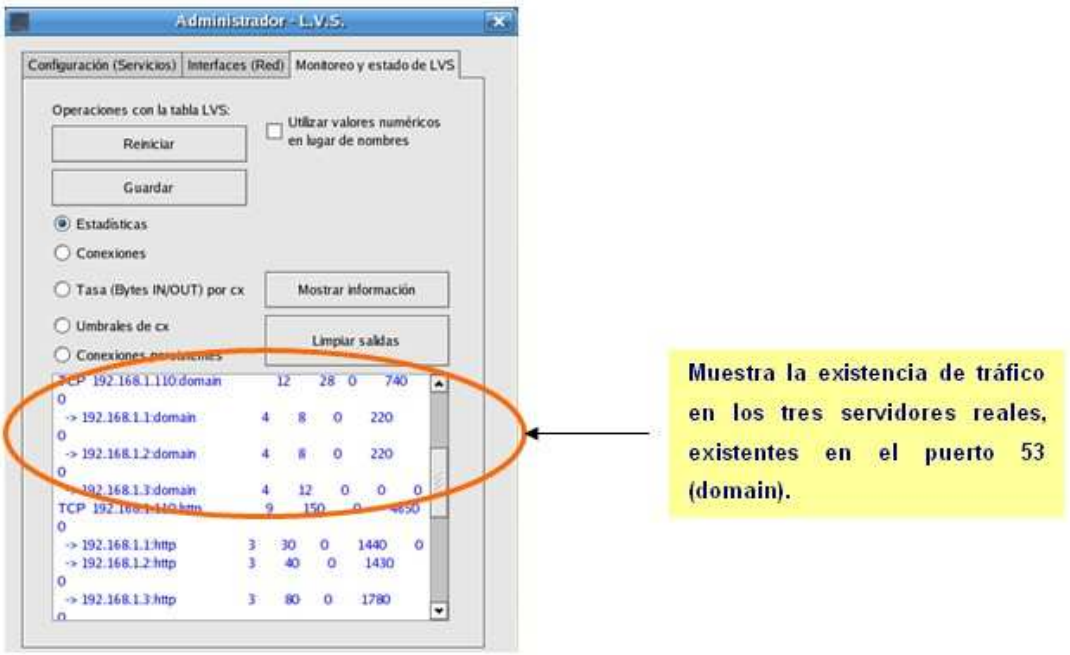


Figura 3-50: Trafico Existente en el puerto 53

3.5.1.4.2 Pruebas de correo electrónico y Web

Para corroborar el funcionamiento de correo electrónico, se lo hizo mediante la utilización de una aplicación Web, la cual permite enviar y recibir correo

electrónico a los usuarios. También se realizó las pruebas de correo electrónico utilizando otro MUA (*Mail User Agent*) como *Microsoft Outlook.*

La Figura 3-51 muestra la utilización de correo electrónico mediante la aplicación Web creada.

Figura 3-51. Ingreso a la aplicación Web para envío de correo.

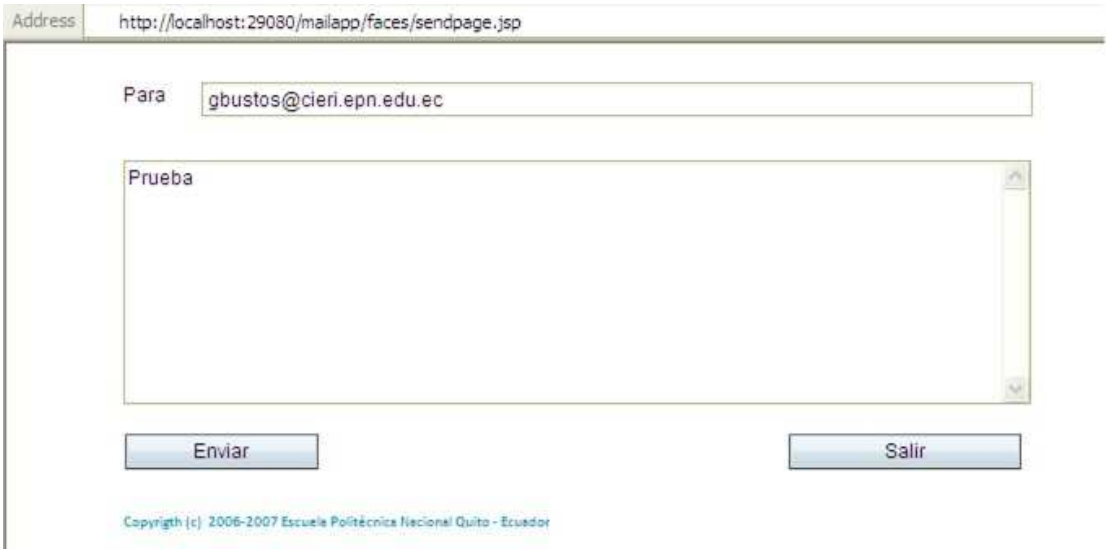


Figura 3-52. Aplicación Web para enviar correos electrónicos y probar la funcionalidad del *cluster*.

La aplicación Web de correo electrónico genera tráfico http a los servidores reales, la Figura 3-53 muestra la existencia de este tráfico en puerto 80 de los servidores.

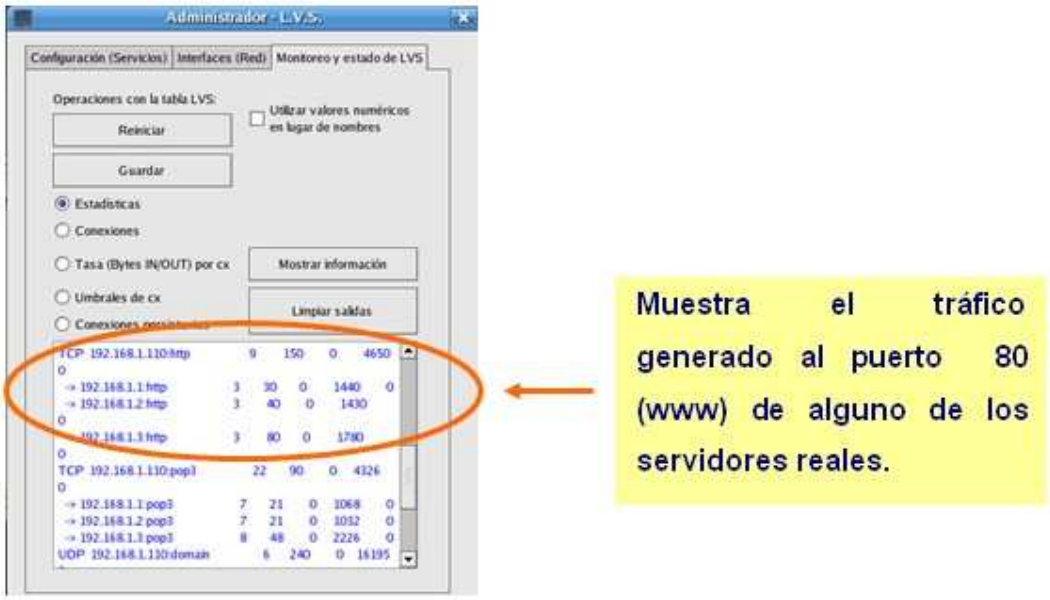


Figura 3-53: Tráfico existente en el puerto 80

Además se configuró el cliente con Microsoft Outlook, la Figura 3-54 indica las pruebas de envío y recepción de correos.

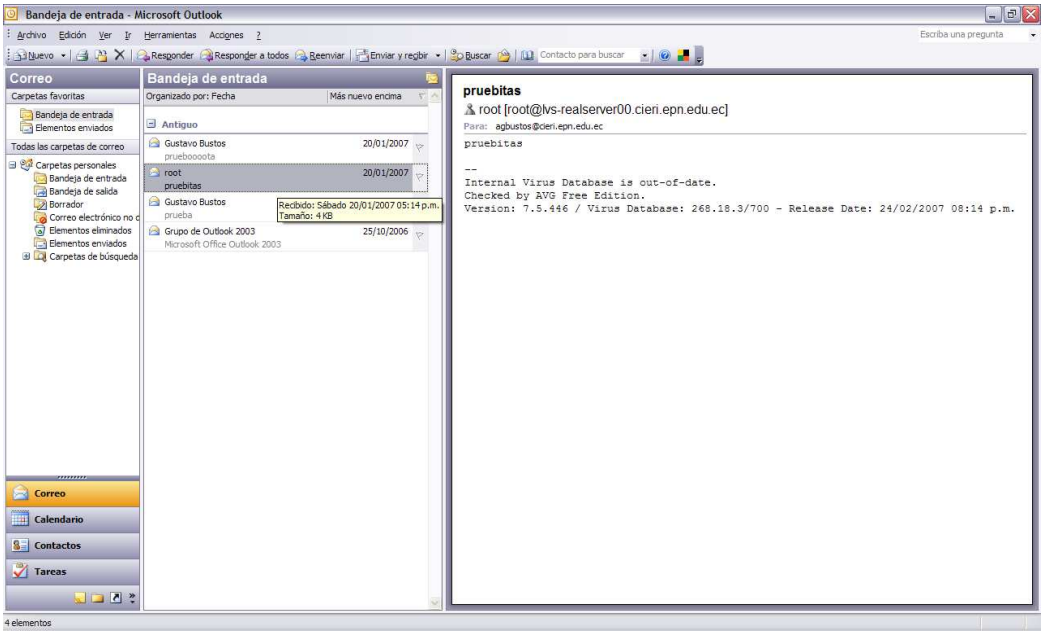


Figura 3-54: Pruebas de funcionamiento

Las pruebas realizadas con los dos MUA (*Microsoft Outlook y* aplicación Web generada) se los realizó al mismo tiempo y éstos en el monitoreador de LVS generaron los siguientes resultados de conexiones en los servidores reales configurados, ver Figura 3-55 y Figura 3-56.

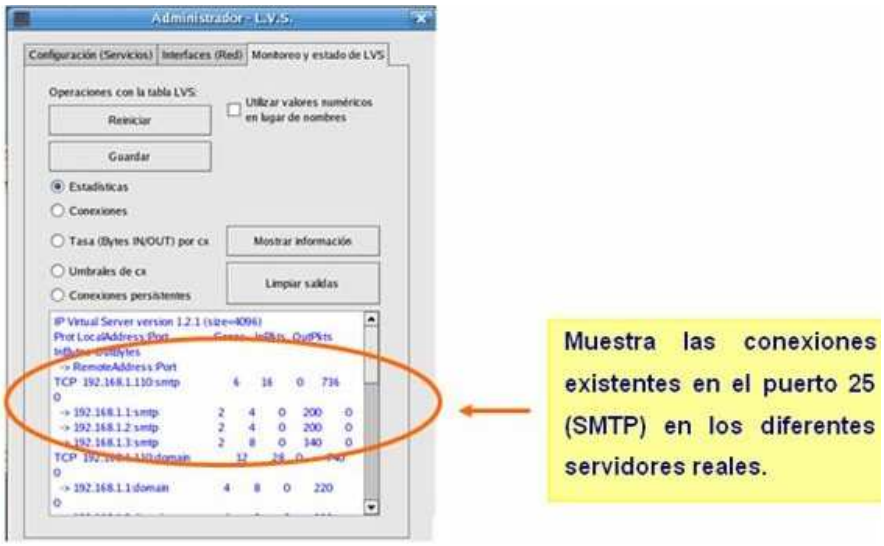


Figura 3-55: Conexiones existentes en el puerto 25

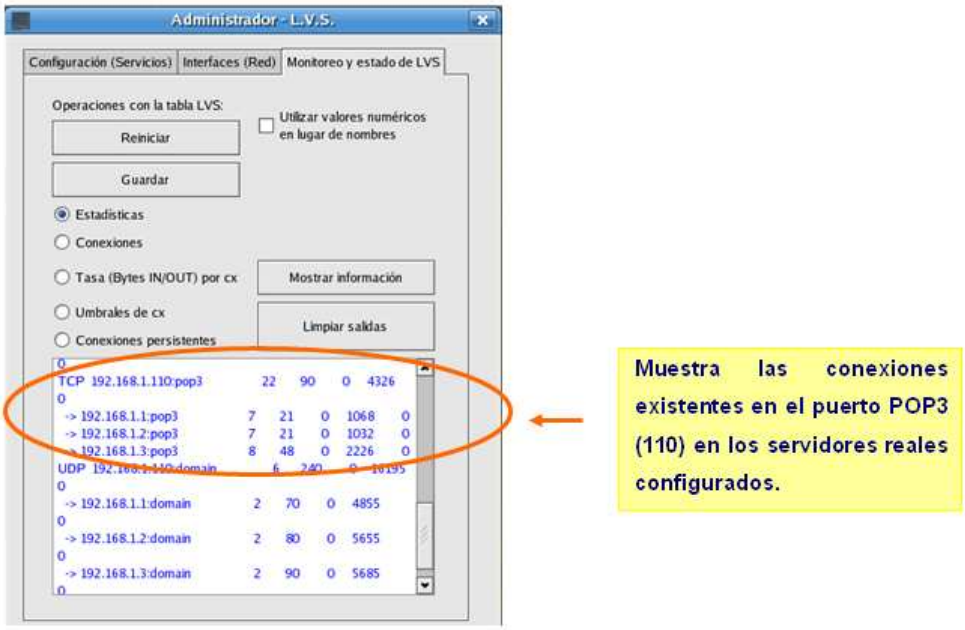


Figura 3-56: Conexiones existentes en el puerto 110

El monitoreador de LVS presenta también las conexiones establecidas entre los clientes, y los servidores reales a través de la dirección IP virtual y su respectivo puerto de servicio. La Figura 3-57 muestra algunas de las conexiones establecidas durante las pruebas realizadas.

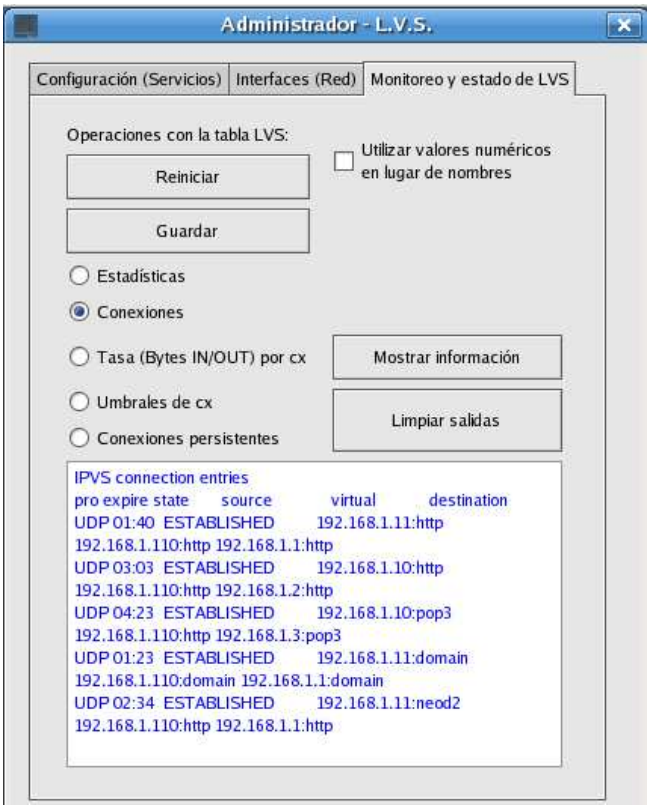


Figura 3-57: Conexiones establecidas

Como parte de la prueba también se quitó uno se los servidores reales (192.168.1.19 con la finalidad de poder comprobar que el servicio se mantiene disponible con los otros servidores reales configurados.

La Figura 3-58 muestra la actividad de conexiones smtp de dos servidores reales menos del servidor real 1.

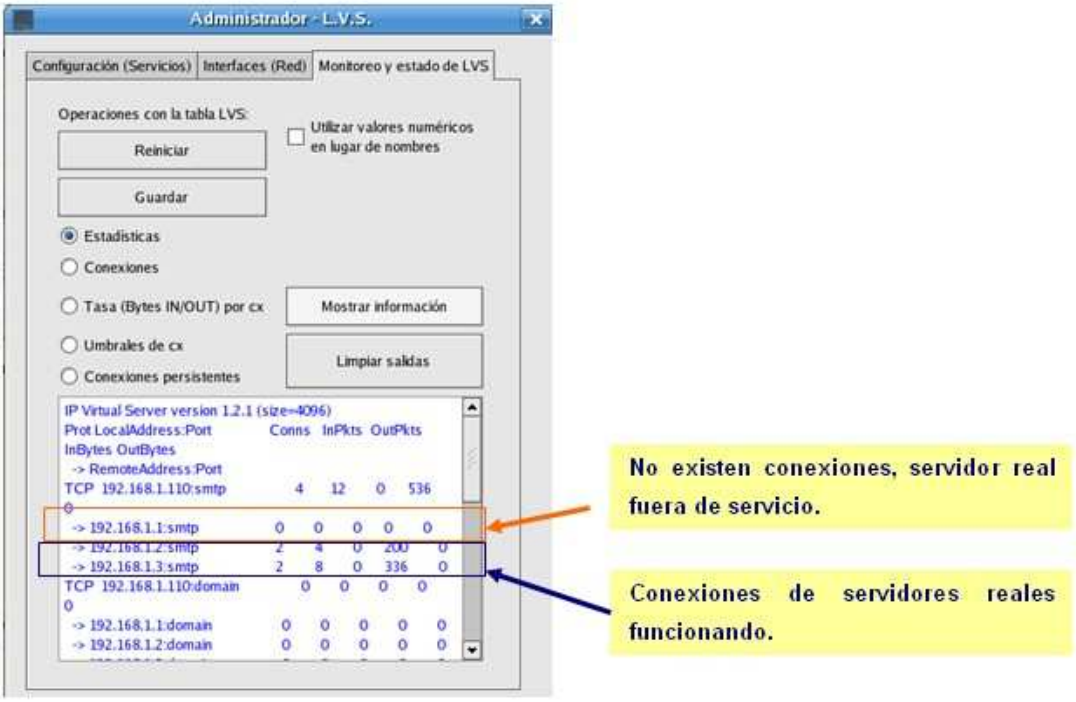


Figura 3-58: Estados de Conexión activos e inactivos

Una vez culminadas las pruebas del *cluster* de alta disponibilidad y balanceo de carga se pudo constatar que el sistema está cumpliendo la funcionalidad para la cual fue creado, se generó el alto tráfico con *ECperf* hacia los puertos de los servicios ofrecidos y se pudo visualizar mediante el monitoreador de LVS que todos respondían, e incluso la carga generada fue distribuida a los servidores reales mediante la técnica *Round Robin*.

También se pudo constatar que los servidores reales configurados pueden suplir la ausencia de uno de ellos, y evitar con ello un punto de falla único. Esto se realizo aislando un servidor real configurado y se pudo verificar que el resto de servidores reales recibían constantemente conexiones generadas desde los clientes comprobando con ello la gran utilidad que presta la granja de servidores configurada en un *cluster* de alta disponibilidad.

La funcionalidad de *HA- OSCAR* también se pudo comprobar, ya que se procedió a sacar manualmente de la red al nodo principal con la finalidad de que el nodo

standby entre en funcionamiento, éste luego de monitorear al nodo principal mediante el enlace de *hertbeat* y el demonio *mon* pudo constatar la no existencia del nodo principal, con lo cual procedió a suplantar su identidad y tomar el control de funcionamiento del *cluster* en general.

Con todas las pruebas del caso realizadas se concluye que el *cluster* de alta disponibilidad y balanceo de carga cumple con las funciones para las que fue creado.

3.6 COMPARACIÓN DE LOS RESULTADOS OBTENIDOS CON OTRAS TECNOLOGÍAS DE SERVIDORES WEB, CORREO ELECTRÓNICO, DE RESOLUCIÓN DE NOMBRES, TECNOLOGÍAS SAN (STORAGE AREA NETWORK) Y NAS (NETWORK ATTACHED STORAGE).

Los resultados obtenidos en un *cluster* de alta disponibilidad y balanceo de carga utilizando el sistema operativo Linux y software de uso libre, arrojó los resultados esperados, es decir se logró tener el balanceo de carga y alta disponibilidad en los servidores reales y directores, así como tener un buen desempeño ante la existencia de un alto tráfico que genera distintos requerimientos al *cluster .*

La forma como se realizaron las pruebas se detallan a continuación, donde la Figura 3-59 indica la manera como se las realizó. Como se puede visualizar existen los servidores reales y los nodos directores, todos ellos utilizando el mecanismo de balanceo de carga basado en enrutamiento directo. El cliente es el encargado de generar los requerimientos en los diferentes servicios (DNS, SMTP, POP3, WWW) y mediante el *benchmark* Ecperf simular alto tráfico hacia los servicios ofrecidos por el *cluster* en general, además se tiene un sistema de archivos distribuido NFS el cual permite tener la información disponible todo el tiempo. Los resultados obtenidos, se muestran en la sección 3.5.1 del presente proyecto.

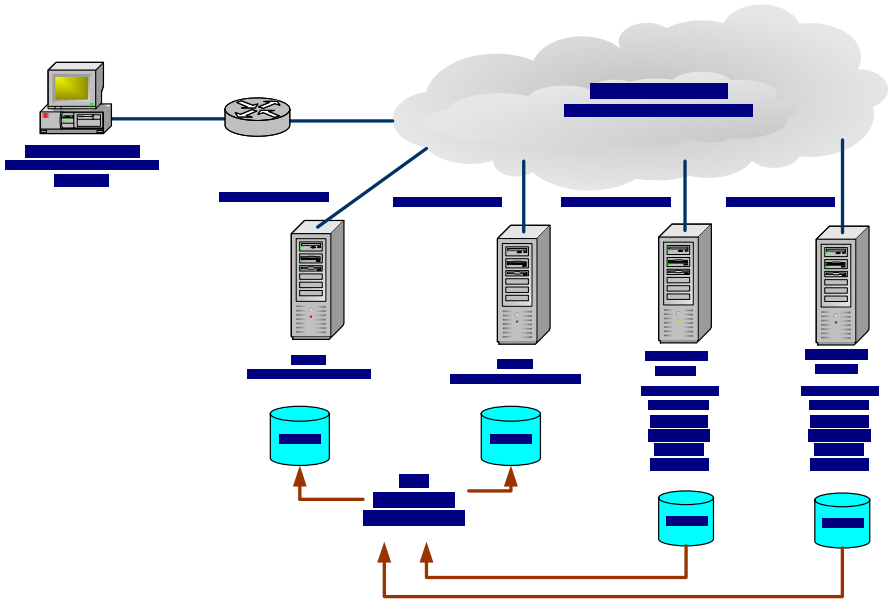


Figura 3-59: Arquitectura de pruebas

El objetivo principal para comparar la tecnología *clustering* de balanceo de carga y alta disponibilidad, con otras tecnologías como SAN y NAS es de verificar el correcto funcionamiento del sistema ante la existencia de un punto de falla en cualquier circunstancia (alta disponibilidad), la manera de distribuir la carga entrante (balanceo de carga).

Con la finalidad de mostrar el desempeño de la tecnología SAN y NAS se hizo un esfuerzo por conseguir los permisos necesarios para realizar las pruebas (pruebas realizadas en el *cluster* de alta disponibilidad y balanceo de carga, es decir configurar servicios de resolución de nombres, Web, correo electrónico y ver el comportamiento de estas tecnologías ante un alto tráfico) sobre equipos de esta naturaleza, lamentablemente, debido al hecho de que son dispositivos cuyo costo es muy elevado y que la existencia de estos es muy limitada en el país, no fue posible tener acceso a éstos y por ende poder realizar las pruebas en los mismos. Por tal motivo no se puede realizar la comparación de funcionamiento en tiempo real entre el *cluster* implementado y las tecnologías SAN y NAS.

Por los motivos antes expuestos, la comparación de resultados obtenidos se realizará en función de mecanismos de alta disponibilidad y balanceo de carga entre el *cluster* implementado y los datos proporcionados por los fabricantes de tecnologías SAN y NAS, que para el efecto se tomo como referencia a IBM.

La Tabla 3-9 muestra una comparación de disponibilidad y balanceo de carga entre las tecnologías mencionadas.

Tecnología	**Mecanismo de alta disponibilidad y balanceo de carga**	**Ventajas**	**Desventajas**
NAS	• Hardware redundante. • Utilización de configuración RAID (0, 1, 3, 5, 10) para garantizar la integridad de datos. • Fuentes de poder redundantes. • varios discos extraíbles y controladoras duales. • Balanceo de carga disponible mediante la utilización de equipos adicionales como *Load-Balancing Internet Servers*	• Múltiples métodos de backup basado en discos y en el host. • Alto nivel de procesamiento de servidores, • Alta tasa de transferencia de datos. • Gran capacidad de almacenamiento. • Soporta varios sistemas operativos (Linux, Windows, *Veritas*, *VMware*). • Conectividad de dispositivos basada en fibra óptica con lo cual se tiene una alta transferencia de datos.	• Alto costo de implementación. • Software licenciado. • Problemas de escalabilidad debido al alto costo de equipos que manejan esta tecnología. • Problemas de costo cuando un determinado servidor esta saturado, es decir cumplió sus funciones, y hay que cambiarlo.

Tecnología	Mecanismo de alta disponibilidad y balanceo de carga	Ventajas	Desventajas
Clustering	• Hardware redundante gracias a la existencia de servidores reales los cuales poseen la misma configuración. • Balanceo de carga mediante la utilización de LVS con lo cual la carga recibida por el nodo principal es repartida a los servidores reales. • El paquete *HA OSCAR* brinda grandes ventajas por el hecho de tener demonios a nivel del sistema operativo dedicados a detectar puntos de falla en el sistema.	• Evita tener un único punto de falla a nivel de servidores ya que la falla de uno de ellos puede ser suplida por el resto de servidores reales. • Alto nivel de escalabilidad y confiabilidad de datos. • Bajo costo de implementación. • Alto nivel de procesamiento a bajo costo. • Software libre para implementación y administración del sistema.	• Baja capacidad de almacenamiento. • Falta de documentación técnica disponible. • Punto de falla único en dispositivos de interconectividad • Transferencia de datos limitada al procesamiento de cada servidor. • Sistema puede ser implementado únicamente en sistemas operativos Linux. • Diferencia en tiempos de respuesta con transferencia de datos utilizando cobre como medio de transmisión con relación a transferencia mediante utilización de fibra óptica.

Tecnología	Mecanismo de alta disponibilidad y balanceo de carga	Ventajas	Desventajas
SAN	• Canales de fibra redundantes. • Fuentes de poder redundantes. • Dispositivos de interconectividad de fibra óptica redundantes. • RAID para integridad de datos almacenados. • Discos extraíbles y controladores duales. • Balanceo de carga disponible mediante la utilización de equipos adicionales como *Load-Balancing Internet Servers.* • *Backup's* de datos mediante la utilización de cintas magnéticas.	• Alta tasa de transferencia de datos gracias a la utilización de la tecnología *fiber chanel*. • Alto nivel de procesamiento a nivel de servidores. • Gran capacidad de almacenamiento. • Soporte técnico disponible. • Mecanismos de *backup's* basados en cintas. • Soporta varios sistemas operativos (Linux, Windows, *Veritas*, *VMware*). • Tasa de transferencia de datos alta.	• Alto costo de implementación y administración del sistema. • Dificultad de escalabilidad debido a altos costos de equipos que manejan esta tecnología. • Software licenciado. • Alto costo de equipos adicionales para balanceo de carga. • Dificultad en cuanto a costo cuando un servidor se encuentra saturado, razón por la cual para solucionar se lo debe cambiar por uno más potente.

Tabla 3-9: Comparación De alta disponibilidad y balanceo de carga [9]

Una vez realizado un estudio de los mecanismos para ofrecer alta disponibilidad y balanceo de carga de las diferentes tecnologías analizadas, se puede llegar a la conclusión que cualquiera de ellas es útil para solucionar estos requerimientos.

Es importante hacer notar algunos aspectos a considerar, con relación al costo de implementación de un *cluster* con computadores personales y software libre es relativamente bajo con relación a tecnologías como SAN y NAS.

Es importante dejar en claro que el funcionamiento y desempeño de equipos con tecnologías SAN y NAS es superior al que presenta el *cluster* implementado, ya que por ser infraestructuras propietarias siempre buscan estar actualizados tecnológicamente, es decir tienen equipo humano especializados para desarrollar nuevas mejoras en sus equipos y con ello tener una adecuada tecnología de punta.

Bibliografía capítulo 3

1. http://www.google.com/technology/pigeonrank.html, *" The technology behind Google's great results"*, última modificación Abril – 2002.
2. http://www.valinux.co.jp/en/, "*VA LINUX",* 2004 – 2007.
3. http://www.steeleye.com/products/linux/, "*LifeKeeper for Linux"*.
4. http://www.missioncriticallinux.com, "*Mission Critical Linux".*
5. http://www.linuxvirtualserver.org/software/index.html, "*Linux Virtual Server – Software"*, creado en 28 de mayo de 1998.
6. http://www.redes-linux.com/manuales/Servidor_correo/configuracion _postfix.pdf, "*Documentación de Postfix".*
7. http://linuxsilo.net/articles/bind.html, "*El sistema de nombres de dominio: BIND 9.2.4"*.
8. www.vmware.com
9. ftp://ftp.software.ibm.com/common/ssi/rep_sp/n/TSO00364USEN/TSO00364USEN.PDF, *"IBM System Storage Product Guide"*.

CAPÍTULO 4
CONCLUSIONES Y RECOMENDACIONES

4.1 CONCLUSIONES

- Una vez realizado el estudio de la tecnología *cluster* se puede concluir la importancia que tiene el software libre en su configuración e instalación, las herramientas y aplicativos que el sistema operativo Linux ofrece lo hacen más atractivo, en especial la posibilidad de manipular su *kernel* para añadir o hacer uso de los diferentes paquetes que se requieren para el funcionamiento de aplicativos y servicios que se este brindando, así como la conveniencia en cuanto a costo ya que utilizar otras herramientas propietarias involucra un gasto adicional.

- Después de estudiar la tecnología *clustering* se puede mencionar la importancia que tiene en ambientes donde se requiere un incremento del número de transacciones o velocidad de respuesta, ofrecido por el *cluster* de balanceo de carga, incremento de confiabilidad y robustez ofrecido por el *cluster* de alta disponibilidad. Estas características son de fundamental importancia en el funcionamiento de un sistema de gran demanda Web, donde el rápido crecimiento del Internet, incremento de comercio electrónico, y otras aplicaciones Web requieren un adecuado tiempo de respuesta y en especial una adecuada disponibilidad de los servicio ofrecidos, estos requerimientos son muy importantes para los usuarios y empresas que ofrecen estos servicios, ya que la indisponibilidad y lentitud en el sistema implicaría pérdidas económicas considerables.

- Una de las principales ventajas de tener un *cluster* es la escalabilidad que éste ofrece, ya que dependiendo de las necesidades de procesamiento permite incrementar servidores reales sin tener un límite establecido, ofreciendo una gran ventaja en comparación a otras tecnologías como supercomputadores, que poseen un gran funcionamiento pero cuando las

necesidades incrementan se encuentran limitados debido a que saturan su capacidad de cálculo y para lograr escalabilidad se debe cambiarlo por uno más potente. La escalabilidad en cuanto a costo es más conveniente cuando se tiene un *cluster* con máquinas personales y software libre, ya que es más conveniente invertir en un servidor de estas características que en un supercomputador el cual es más costoso.

- La alta disponibilidad en un ambiente de gran demanda Web y resolución de nombres tiene mucha importancia en diferentes aspectos, uno de ellos es la facilidad de realizar el mantenimiento de servidores. Una máquina de un *cluster* de servidores se puede sacar de línea, apagarse y actualizarse o repararse sin comprometer los servicios que brinda el cluster y sin afectar el desempeño del sistema en general.

- Luego de la instalación y configuración del *cluster* de balanceo de carga utilizando *Linux Virtual Server* LVS, se concluye que es una herramienta de mucha utilidad que proporciona ayuda para cualquier tipo de ambiente sea este homogéneo o heterogéneo, debido a que el paquete IPVS que viene incluido en la versión de *kernel* 2.6 posee diversas opciones que fueron configuradas con la herramienta *ipvsadm,* la misma que es de uso libre y que permite configurar el *cluster* en su totalidad.

- Para el proceso de configuración de alta disponibilidad se escogió *HA OSCAR* herramienta que posee un ambiente gráfico de instalación, con el cual se tuvieron algunos inconvenientes al momento de habilitar el demonio *mon* para balizar el monitoreo*,* pues al momento de activar este demonio existía problemas en su ejecución ya que no encontraba los módulos *perl_time_period y perl_convert_BER* estos módulos son de mucha importancia para el proceso de monitoreo, es por ello que es de vital importancia mencionarlos para que se tomen en cuenta ya que *HA-OSCAR* como tal no los trae incorporados.

- Durante el proceso de instalación de LVS se tuvo inconvenientes debido a conflictos a nivel de ARP, ya que la interfaz virtual del *cluster* estaba configurada en todos los servidores, sean estos reales o director, para solucionar el problema se tuvo que aplicar un parche en los servidores reales y recompilar el *kernel* para que este tenga efecto, con esto se logró solucionar el problema, siendo el nodo director el único que tiene la dirección IP virtual expuesta al cliente y quien redirecciona el trabajo a los servidores reales.

- Los mecanismos de balanceo de carga existentes ofrecen un adecuado funcionamiento, pero es muy importante tomar en cuenta diversos aspectos antes de escoger cual de ellos utilizar. En el presente caso de estudio se decidió utilizar el mecanismo de enrutamiento directo, debido a que las características de hardware del nodo director primario y *standbye* no eran las adecuadas para realizar traslación de direcciones (balanceo por NAT) y encapsulamiento IP, los cuales con un alto tráfico necesitan gran capacidad de cálculo para poder trabajar de manera similar al utilizado por enrutamiento directo.

- De la pruebas realizadas en el *cluster* de balanceo de carga se pudo constatar que la granja de servidores ubicada tras lo nodos directores ofrecen gran desempeño al momento de dar un determinado servicio, ya que por el hecho de existir varios servidores reales configurados de idéntica manera permiten tener un mayor número de conexiones simultáneas, y con ello tener mayor eficiencia de funcionamiento del *cluster*, la diferencia de funcionamiento se hizo notoria cuando se realizó las pruebas con dos servidores reales configurados pero uno de ellos inactivo, el número de conexiones en un tiempo determinado disminuyó y el tiempo de respuesta aumentó con relación al grupo de servidores reales ofreciendo el mismo servicio.

- De las prueba realizadas con *HA-OSCAR* se pudo constar la utilidad que tiene el demonio *mon* ya que permite detectar cuando el nodo director

principal queda fuera de funcionamiento, además el funcionamiento del enlace de *heartbeat* es el deseado ya que mediante este se realiza el proceso de suplantación de identidad del servidor director al *standby* en caso de la existencia de una falla.

- Con relación a la comparación de resultados de la tecnología *cluster* con otras tecnologías como SAN y NAS se realizó la comparación de los mecanismos de alta disponibilidad y balanceo de carga que éstas poseen, existiendo la dificultad de no poder realizar las pruebas de funcionamiento debido a que son equipos muy costosos y la carencia de posibilidades de acceder a lugares donde están implementados.

4.2 RECOMENDACIONES

- Se recomienda tener alta disponibilidad a nivel de servidores y a nivel de dispositivos de interconectividad como *routers* y *switch,* ya que si éstos se convierten en un punto de falla único, no sirve de nada tener alta disponibilidad en el resto de dispositivos.

- Se recomienda realizar la configuración utilizando el proyecto Linux HA, el cual es una herramienta completa que se acopla de manera adecuada para trabajar en conjunto con LVS y puede ser administrada de manera sencilla con el paquete *webmin*.

- Para tener un adecuado funcionamiento en los servidores principal y *standby* es adecuado que posean infraestructura homogénea, ya que el rendimiento de ambos en caso de falla debe ser el mismo y con ello lograr tener un funcionamiento uniforme de tal manera que el *cluster* no se vea afectado con la operación de cualquiera de ellos.

- Se considera importante estudiar el comportamiento de los usuarios en el país respecto al Internet y del tráfico que éstos generan, con la finalidad de contar con información sobre las necesidades de estos usuarios y poder desarrollar aplicaciones y servicios que las satisfagan.

- La importancia de tener gran capacidad computacional en aplicaciones a nivel empresarial es fundamental, pero muchas veces el excesivo costo de los equipos es una limitante, para solucionar este inconveniente se debería dar una utilidad adecuada a equipos personales que no se estén utilizando o se consideren obsoletos, con los cuales se puede construir un *cluster* que solucione problemas de alto procesamiento a un costo bajo.

ANEXO A

Comandos Ipvsadm

Ipvsadm para su funcionamiento tiene dos formatos básicos de ejecución:

- **ipvsadm** COMANDO [Protocolo] [dirección IP de servicio virtual][: Puerto] [tipos de balanceo de carga] [Opciones relacionadas a conexiones persistentes]

- **ipvsadm** comando [Protocolo] [dirección IP de servicio virtual] [: Puerto] [dirección IP de servidor real] [: Puerto] [mecanismo de balanceo de carga] [opciones de peso]

COMANDO. Ipvsadm reconoce las órdenes que se especifican a continuación, aquellas órdenes detalladas con mayúscula hacen referencia a los servicios virtuales.

- -A (añadir servicio). Agrega un servicio virtual. Para ello se debe asignar una dirección de servicio la cual esta compuesta por una dirección IP, número del puerto, y un protocolo.

  ipvsadm -A –t (tcp) |u (udp) dirección IP de servicio virtual [-s tipos de balanceo de carga] [-p [*timeout*]] [-M mascara de red]

- -E (edición de servicio). Utilizado para editar el servicio virtual.

  ipvsadm -E –t (tcp) |u (udp) dirección IP de servicio virtual [-s tipos de balanceo de carga] [-p [*timeout*]] [-M mascara de red]

- -D (borrado de un servicio). Parámetro que se encarga de borrar un servicio virtual, junto con cualquier servidor real asociado.

ipvsadm -D –t (tcp) |u (udp) dirección IP de servicio virtual

- -C (vaciar). Parámetro que se encarga de vaciar la tabla del servidor virtual.

  ipvsadm -C

- -R (restaurar). Permite restaurar las reglas de LVS desde *stdin*, cada línea leída desde el *stdin* debe ser tratada como una línea de comandos.

  ipvsadm -R

- -S (guardar). Descarga las reglas del Linux virtual Server al *stdout* en un formato que pueda ser leída por -R (restaurar).

  ipvsadm -S [-n]

- -L, (listar). Permite listar la tabla de servidores virtuales si no se especifican argumentos, en caso de listar el argumento –c, lista la tabla de conexiones existentes con el servicio virtual.

  ipvsadm –L [opciones]

- -Z, (*zero*). Contadores de un servicio del sistema son puestos en cero.

  ipvsadm -Z [-t (tcp) |u (udp) dirección IP del servicio virtual]

comando. Opciones especificadas con minúscula hacen referencia a los servidores reales asociados con un servicio virtual.

- -a, (añadir servidor real). Parámetro que añade un servidor real al servicio virtual.

ipvsadm -a –t (tcp) |u (udp) [dirección IP de servicio virtual] [-r dirección IP de servidor real] [-g (enrutamiento directo)|i (encapsulamiento ip) |m (NAT)] [-w (opciones de peso)]

- -e, (editar un servidor). Permite editar un servidor real del servicio virtual.

  ipvsadm -e –t (tcp) |u (udp) [dirección IP de servicio virtual] [-r dirección IP de servidor real] [-g (enrutamiento directo)|i (encapsulamiento ip) |m (NAT)] [-w (opciones de peso)]

- -d, (remueve un servidor real). Permite remover un servidor real del servicio virtual.

  ipvsadm -d –t (tcp) |u (udp) dirección IP de servicio virtual] [-r dirección IP de servidor real]

- -l, (listar). Permite listar la tabla de servidores virtuales si no se especifican argumentos, en caso de listar el argumento –c, lista la tabla de conexiones existentes con el servicio virtual.

  ipvsadm –l [opciones]

Protocolo. Protocolos que pueden ser configurados en el *cluster* de balanceo de carga, los cuales se detallan a continuación:

- -t, (-- servicio tcp – dirección de servicio). Este parámetro es utilizado cuando se está ofreciendo servicios *TCP*, la dirección de servicio tiene la forma *host* [: puerto], *host* puede ser una dirección *IP* del servicio o un *hostname*. El puerto debe ser especificado de acuerdo al tipo de servicio que se este utilizando. El puerto puede ser omitido en el caso de estar utilizando servicios persistentes, en este caso el puerto toma un valor de cero y en este caso las conexiones deben ser aceptadas desde cualquier puerto.

- -u, (-- servicio udp – dirección de servicio). La descripción de este parámetro es la misma que la utilizada en servicios tcp.

Tipos de balanceo de carga. Los diferentes tipos de balanceo de carga son utilizados en conexiones TCP y UDP, los cuales son implementados como módulos del *kernel*, a continuación se describen los parámetros de configuración de estos tipos de balanceo, la descripción del funcionamiento de estos se la realizó ya en el capítulo 2 ítem 2.2.2.5.

- rr, (*Round Robin*).
- wrr, (*Weighted Round Robin*). *Round Robin* ponderado.
- lc, (*Least-Connection*). Menos conexiones activas.
- wlc, (*Weighted Least-Connection*). Menos conexiones activas ponderadas.
- lblc, (*Locality-Based Least-Connection*). Menos conectado basado en servicio.
- lblcr, (Localidad basada en menos conexiones con replicación).
- sh, (fuente *hashing*).
- sed, (*Shortest Expected Delay*).
- nq, (ninguna cola).

Conexiones persistentes. Existen diversas opciones de configuración las cuales se detallan a continuación:

- -p, --*persistent*[*timeout*]. Parámetro que permite especificar que servicio virtual es persistente, múltiples requerimientos desde un cliente son redireccionados para el mismo servidor real seleccionado en el primer requerimiento. Opcionalmente, un *timeout* de sesiones persistentes debe ser especificado, por defecto se asignan 300 segundos. Esta opción debería ser usada en conjunto con protocolos SSH (*Secure Socket Layer*) o FTP donde es importante que los clientes se conecten constantemente con el mismo servidor.

- -M, (-- *netmask netmask*). Especifica que los clientes son agrupados para servicios virtuales persistentes. La dirección fuente de un requerimiento es enmascarada con esta mascara de red para direccional todos los clientes desde una red a un mismo servidor real. La mascara de red por defecto es 255.255.255.255.

- -r, (-- real-server server-address). Este parámetro permite asignar una dirección IP a un servidor real, además se debe añadir el puerto que indica el tipo de servicio que se esta ofreciendo.

Mecanismos de balanceo de carga. Dentro de los mecanismos de balanceo de carga existen tres tipos principales y algunas opciones adicionales que se detallan a continuación:

- -g, (-- *gatewaying*). Este parámetro indica que se esta utilizando el mecanismo de balanceo de carga basado en enrutamiento directo.

- -i, (-- ipip). Parámetro que indica que el mecanismo de balanceo de carga utilizado es el encapsulamiento IP.

- -m, (-- masquerading). Método de balanceo de carga basado en NAT (*network access translation*).

- -w, (-- *weight weight*). El peso es un entero que especifica la capacidad de cálculo de un servidor relativo a los otros servidores que conforman el *cluster.* Los valores validos de peso están en el rango que va desde 0 hasta 65535. El valor por defecto es 1. Un servidor inactivo esta especificado con un valor de peso igual a 0, tiene la característica de no recibir nuevos trabajos, pero continúa sirviendo a los existentes. La configuración de un servidor inactivo debe ser especificada cuando un servidor se encuentra sobrecargado o cuando se realiza algún mantenimiento.

- -x, (-- u – *threshold uthreshold*). *Uthreshold* es un entero que especifica las conexiones superiores al umbral permitido de un servidor. Los valores validos de *uthreshold* están entre 0 hasta 65535. El valor por defecto es 0, el cual significa que la conexión superior al umbral permitido no esta fijada. Si *uthreshold* esta fijado con otro valor superior al permitido, ninguna conexión nueva deberá ser enviada al servidor.

- -y, (--l - *threshold lthreshold*). *lthreshold* es un entero que especifica las conexiones menores al umbral permitido de un servidor. Los valores validos de *lthreshold* están entre 0 hasta 65535. El valor por defecto es 0, el cual significa que la conexión menor al umbral permitido no esta fijada.

- -c, --connection. Conexiones de salida. Utilizado con el comando que permite listar conexiones IPVS.

- --stats, (estadísticas). La lista de comando que utilizan esta opción, desplegaran información de estadísticas de los servicios y sus servidores.

- --rate (tasa de salida de datos). La lista de comandos que utilizan esta opción deberán mostrar la tasa de información mostrada en conexiones/segundo, bytes/segundo y paquetes/segundo de los servicios y sus servidores.

- --thresholds. (umbrales). La lista de comandos con esta opción mostrara información de la conexión de *threshold* (umbral) mas alta y mas baja de cada servidor en la lista de servicios

- --persistent-conn.(conexiones persistentes) La lista de comandos con esta opción deberá mostrar información relacionada a las conexiones persistentes para cada servidor en la lista de servicios. Las conexiones persistentes son usadas para pasar las actuales conexiones desde el mismo cliente/red al mismo servidor.

- --sort. Clasifica la lista de servicios virtuales y servidores reales. Los servicios virtuales entrantes son clasificados en orden ascendentes por protocolo, dirección y puerto. Los servidores reales se clasifican en orden ascendente por dirección y puerto.

- -n, (--numérico). Las direcciones IP y números de puerto serán impresos en un formato numérico.

ANEXO B

Configuración de DNS

1. En primer lugar se debe ingresar al sistema como administrador ***root.***

2. Se debe modificar el archivo ***named.conf*****.** En el sistema operativo *Fedora* se debe chequear si el paquete ***bind-chroot.RPM*** ha sido instalado, es importante exista, ya que el archivo ***named.conf*** se ejecuta sobre este ambiente, es decir en el directorio **/var/named/chroot/etc/named.conf.** El comando que nos permite visualizar si el paquete *bind-chroot* esta instalado es:

   fedora# **rpm -q bind-chroot**

3. Crear el archivo con el nombre de la zona. Este debería incluir todos los host existentes en el dominio.

   fedora# **vi /var/named/chroot/var/named/named.cieri.epn.edu.ec**

4. Crear el archivo de zona IP, de manera similar a como se lo hizo con el archivo del nombre de zona.

   fedora# **vi /var/named/chroot/var/named/named.192.168.1**

5. Posteriormente se debe verificar los archivos de zona local. Estos archivos deben ser instalados de manera automática el momento de instalar el sistema operativo con el servicio DNS habilitado. Estos archivos se encuentran en los siguientes directorios:

   fedora# **vi /var/named/chroot/var/named/localhost.zone**

En estos directorios se debe especificar el nombre de la zona, sin modificar el puntero al localhost.

6. Copiar el archivo */etc/host* pata crear un *backup*. Luego de ello editar el archivo */etc/host,* eliminando todos los hosts excepto el *localhost* del sistema.

```
fedora# cp /etc/hosts /etc/hosts.bak
fedora# vi /etc/hosts
```

En el archivo */etc/hosts* debe estar únicamente la siguiente información:

```
127.0.0.1       localhost
192.168.1.3   lvsrealserver00.cieri.epn.edu.ec   lvsrealserver00
```

7. Se debe cambiar el nombre del host por el nombre de dominio adecuado.

```
fedora# vi /etc/sysconfig/network
cambiar HOSTNAME: lvs-realsever00.cieri.epn.edu.ec
```

8. Crear el archivo **/etc/resolv.conf**

```
fedora# vi /etc/resolv.conf
```

En el archivo /etc/resolv.conf debe estar únicamente la siguiente información:

```
domain          cieri.epn.edu.ec
nameserver      192.168.1.110
```

9. Se debe reiniciar el demonio *named* y verificar que se inicie sin problemas.

```
fedora/redhat# service named start
```

fedora/redhat# **service named status**

10. Verificar si los archivos de *backup* fueron realmente creados.

    fedora# **cd /var/named/chroot/var/named**
    # **ls -l**
    # **cat lvs-realserver00.cieri.epn.edu.ec.bak**
    # **cat named.192.168.1.bak**

Habilitando un cliente. Los paso que se detallan a continuación se deben realizar únicamente en los clientes mas no en los servidores de nombre.

11. Copiar el archivo */etc/hosts* para crear un *backup*. Luego editar este archivo comentando todas las líneas, excepto las relacionadas al *localhost*. Este paso no es necesario, pero se debe asegurar que se esta utilizando el servidor de nombres exclusivamente.

    fedora# **cp /etc/hosts /etc/hosts.bak**
    fedora# **vi /etc/hosts**

    En el archivo */etc/hosts* debe estar únicamente la siguiente información:

    127.0.0.1 localhost
    192.168.1.2 *nombre del cliente*.cieri.epn.edu.ec nombre del cliente

12. Se debe cambiar el nombre del host por el nombre de dominio adecuado.

    fedora# **vi /etc/sysconfig/network**
    cambiar HOSTNAME: *nombre del servidor*.cieri.epn.edu.ec

13. Crear el archivo **/etc/resolv.conf**

    fedora# **vi /etc/resolv.conf**

En el archivo /etc/resolv.conf debe estar únicamente la siguiente información:

```
domain          cieri.epn.edu.ec
nameserver      192.168.1.110
```

ANEXO C

Configuración de Correo Electrónico Postfix.

Una vez instalado el paquete *postfix* se procede a configurar mencionado servicio:

Configurando los registros DNS MX.

1. En el servidor master, añadir el registro MX al archivo de nombre de zona y reiniciar el servicio *named.*

fedora# **vi /var/named/chroot/var/named/ lvs-realserver00.cieri.epn.edu.ec**

Luego añadir la línea:

IN MX 10 lvs-realserver00.cieri.epn.edu.ec

Posteriormente reiniciar el servicio *named.*

fedora# **service named reload**

2. Para poder comprobar que los cambios tuvieron efecto, usar el comando dig.

fedora# dig@lvs-realserver00.cieri.epn.edu.ec cieri.epn.edu.ec

3. Configurar *postfix* para aceptar correos para este dominio, desde todos los sistemas en la red.

fedora# **vi /etc/postfix/main.cf**

Cambiar las siguientes líneas:

myhostname = lvs-realserver00.cieri.epn.edu.ec cieri.epn.edu.ec
mydomain = cieri.epn.edu.ec
myorigin = $mydomain

inet_interfaces = all
mydestination = $myhostname, localhost.$mydomain, $mydomain

4. Iniciar el servicio *Postfix.*

fedora# rcpostfix restart

Una vez activado y configurado el servicio de *postfix* se debe habilitar el servicio pop3 el cual va ser útil para la descarga del correo de cada cliente.

5. Habilitar el demonio *dovecot* para pop3.

fedora# **vi /etc/dovecot.conf**
protocols = pop3
fedora# **chkconfig dovecot on**
fedora# **service dovecot start**

6. Se debe chequear si el demonio de pop3 esta escuchando en el puerto 110.

fedora# **netstat -antp | grep 110**

Una vez habilitado el protocolo POP3, es fundamental tomar en cuenta que el *password* de POP3 se envía en texto plano, es decir sin ninguna protección. Para ello es fundamental agregarle seguridades mediante el uso de encripción SSL (*Secure Socket Layer*). Para ello se debe utilizar el protocolo POP3 seguro POP3s.

7. Habilitando y probando POP3s. Es importante cerciorarse que el certificado firmado por *dovecot* existe en la localización apropiada. Este certificado fue generado cuando *dovecot* fue instalado. Luego de ello se debe habilitar el demonio *dovecot* para POP3S.

fedora# **ls -l /usr/share/ssl/certs**

Debería listar el archive *dovecot.pem*

fedora# **vi /etc/dovecot.conf**

añadir el texto siguiente en la línea:

protocols = pop3 pop3s

Reiniciar el servicio:

fedora# **service dovecot restart**

8. Posteriormente se debe verificar si el demonio de POP3S esta escuchando en el puerto 995.

fedora# **netstat -antp | grep 995**

Se debería observar el demonio *dovecot* de *Fedora* quien es el que escucha en el demonio 995.

ANEXO D

VMWARE

Sistema desarrollado por *VMware Inc*, filial de la Corporación EMC, la cual es la principal proveedora de software de virtualización para arquitecturas X86. VMware permite realizar un sistema de virtualización vía software, el cual permite ejecutar varios sistemas operativos (computadores) dentro de un mismo hardware de manera simultanea, con lo cual se logra dar mayor utilidad a los recursos de un computador o servidor, pero el rendimiento de estas máquinas virtuales depende de las características de hardware del computador/servidor principal, ya que la virtualización hace uso de los recursos físicos de éste.

Un sistema virtual por software permite simular un sistema físico con sus características de hardware las cuales se las puede especificar el momento de la instalación, la máquina virtual emula poseer su propio CPU, tarjeta gráfica, memoria RAM, tarjeta de red, sistema de sonido, conexión USB, disco duro (pueden ser varios); entre otros dispositivos.

El software de *VMware* puede correr en sistemas operativos Windows, Linux, y puede virtualizar sistemas operativos como Linux, Windows, Sun, Novell; entre otros.

Existen diferentes presentaciones de *VMWARE* las cuales se detallan a continuación.

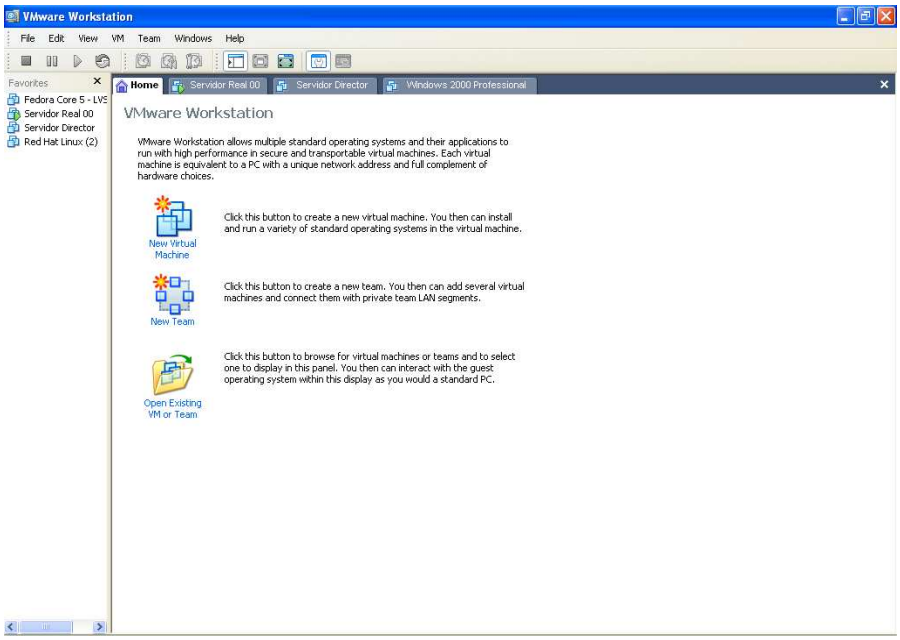
- VMware Server. En un principio era una versión pagada, hace unos meses fue liberada para ser descargada y utilizada de forma gratuita. Esta versión, tiene un mejor manejo y administración de recursos; también corre dentro de un sistema operativo (host), está pensada para responder a una demanda mayor que el Workstation.

- VMware Workstation. Es uno de los más utilizados pues permite la emulación en plataformas PC X86, esto permite que cualquier usuario con una computadora de escritorio o laptop pueda emular tantas máquinas

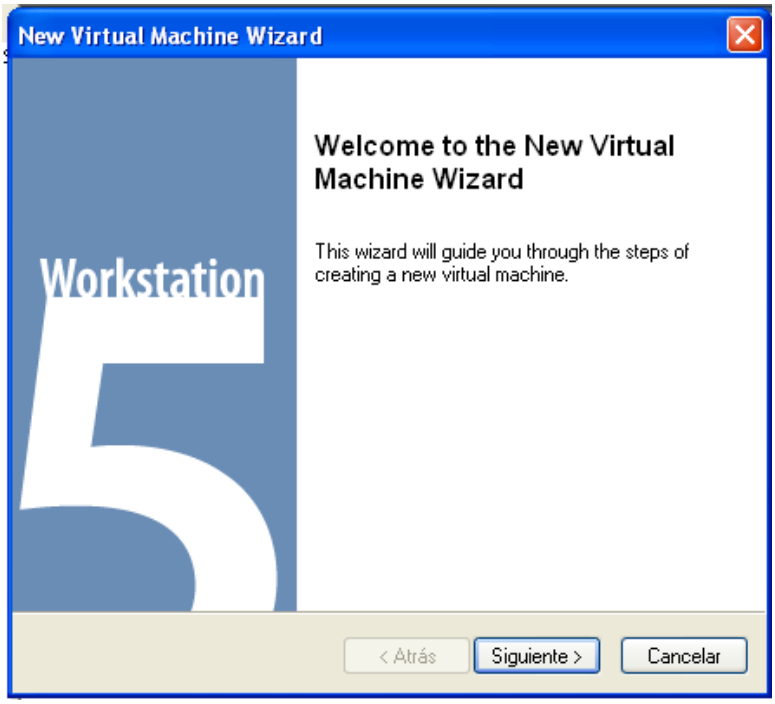
virtuales como los recursos de hardware lo permitan. Esta versión es una aplicación que se instala dentro de un sistema operativo (host) como un programa estándar, de tal forma que las máquinas virtuales corren dentro de esta aplicación, existiendo un aprovechamiento restringido de recursos.

- VMware ESX Server. Esta versión es un sistema complejo de virtualización, pues corre como sistema operativo dedicado al manejo y administración de máquinas virtuales dado que no necesita un sistema operativo host sobre el cual sea necesario instalarlo. Pensado para la centralización y virtualización de servidores, esta versión no es compatible con una gran lista de hardware doméstico.

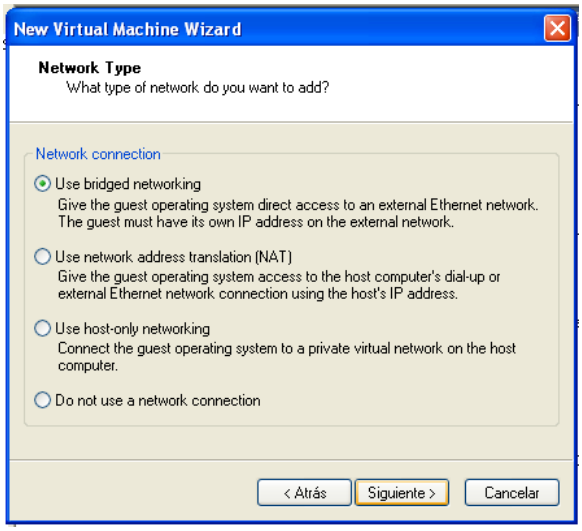
A continuación se describen algunas pantallas donde se detallan los procesos de instalación de una máquina virtual sobre la cual se va instalar cualquier sistema operativo. En la figura se muestra la pantalla principal de *VMware Workstation* donde la primera opción (*create new virtual machine*) es la que permite configurar una nueva máquina virtual.



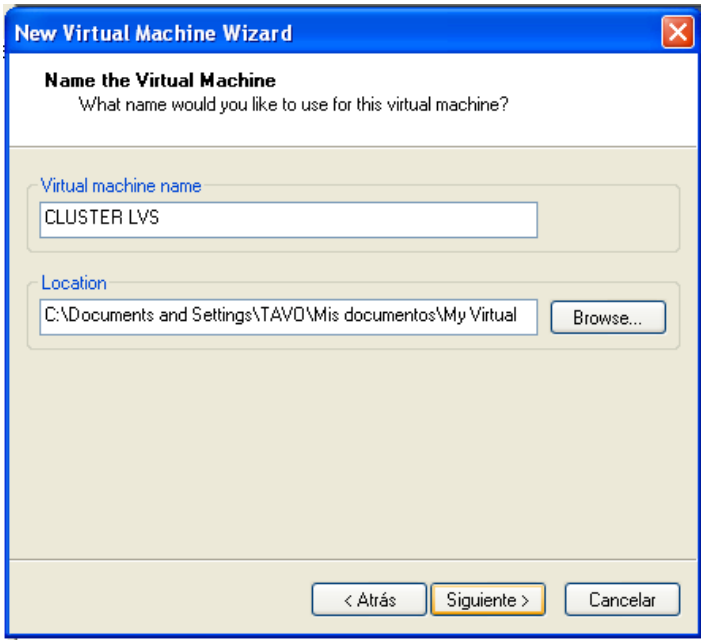
En la figura se muestra la pantalla inicial del proceso de configuración de una máquina virtual, el *wizard* guiará el proceso de instalación.



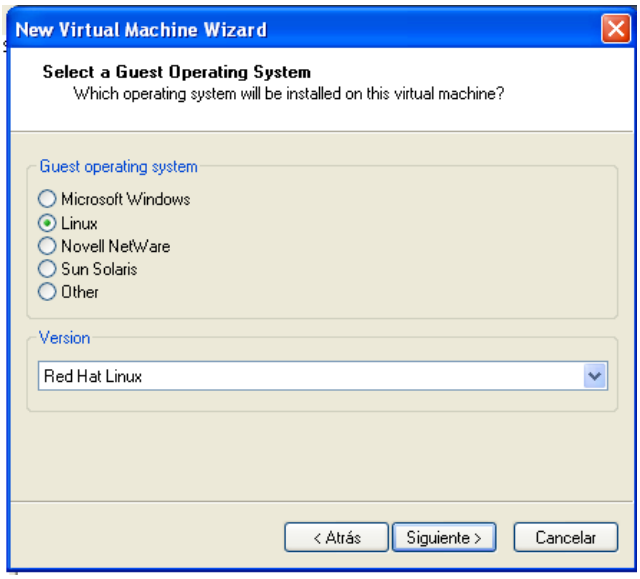
Posteriormente se debe configurar el tipo de conexión de red que se requiere añadir, para el presente caso se necesita que la conexión de red se encuentre hecho *bridge.*



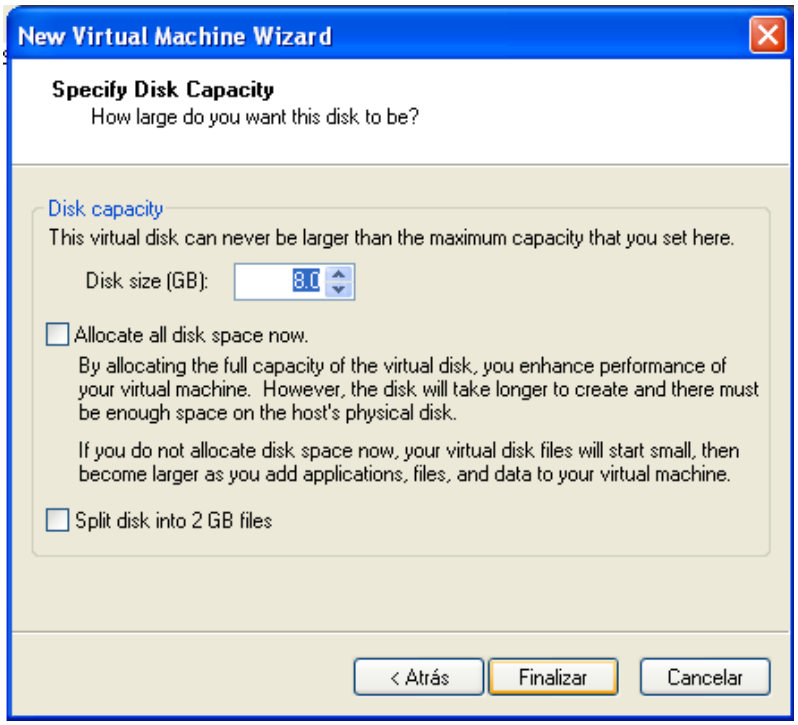
En la figura se debe ingresar el nombre de la máquina virtual y especificar el lugar donde se va instalar.



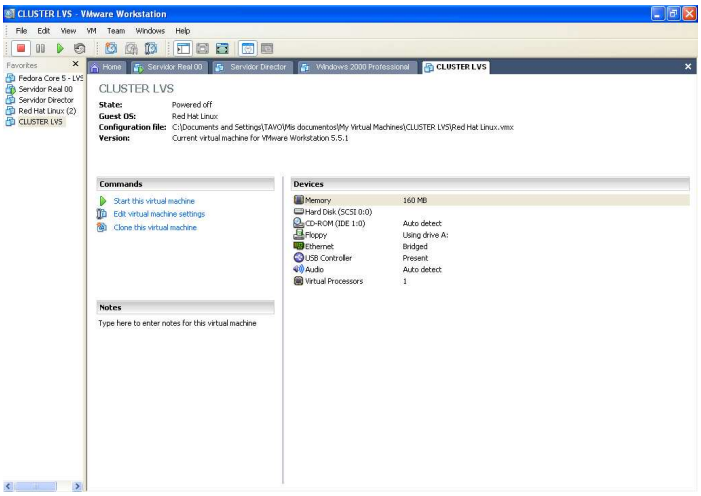
En la figura se debe escoger el tipo de sistema operativo a instalarse, una máquina virtual *VMware* soporta diferentes sistemas operativos los cuales pueden instalarse sobre ésta.



Se debe especificar la capacidad de disco duro que la máquina virtual va utilizar, la figura muestra lo comentando, es importante mencionar que una máquina virtual permite configurar varias unidades de disco virtuales, las cuales se instalan sobre una o varias unidades físicas.



Una vez completa la instalación se presenta la pantalla principal de la máquina virtual configurada, donde se puede observar todos los dispositivos instalados, en la opción *start this virtual machine* hace que la máquina virtual empiece a ejecutarse, es aquí donde podemos empezar la instalación de cualquier sistema operativo.


CLUSTER LVS - VMware Workstation
File Edit View VM Team Windows Help
Favorites
Pedora Core 5 - LVS
Servidor Real 00
Servidor Director
Red Hat Linux (2)
CLUSTER LVS
Home
Servidor Real 00
Servidor Director
Windows 2000 Professional
CLUSTER LVS
CLUSTER LVS
State: Powered off
Guest OS: Red Hat Linux
Configuration file: C:\Documents and Settings\TAYO\Mis documentos\My Virtual Machines\CLUSTER LVS\Red Hat Linux.vmx
Version: Current virtual machine for VMware Workstation 5.5.1
Commands
Start this virtual machine
Edit virtual machine settings
Clone this virtual machine
Devices
Memory 160 MB
Hard Disk (SCSI 0:0)
CD-ROM (IDE 1:0) Auto detect
Floppy Using drive A:
Ethernet Bridged
USB Controller Present
Audio Auto detect
Virtual Processors 1
Notes
Type here to enter notes for this virtual machine