UNIVERSIDAD CATÓLICA DEL MAULE
Facultad de Ciencias de la Ingeniería
Escuela de Ingeniería Civil Informática

Profesor Guía
Dra. Angélica Urrutia
Sr. Marcelo Trujillo

IMPLEMENTACIÓN DE UN SERVIDOR WEB APACHE SOBRE UN CLUSTER EN LINUX

JUAN ESTEBAN CÁCERES VILCHES

CRISTIAN ALEJANDRO MEDINA AMADOR

Seminario de Título para optar al Título de
Ingeniero de Ejecución en Computación e Informática

Talca, Julio 2007

AGRADECIMIENTOS

Damos gracias a Dios por todas las cosas maravillosas que Él ha hecho en nuestras vidas, por encontrarnos en estas instancias, seguros que Él ha estado siempre junto a nosotros.

Al amor y el apoyo, entregado por nuestros padres durante todos estos años; su ayuda y preocupación, en nuestra formación académica, por que siempre han estado presentes.

A la Dra. Angélica Urrutia, por su ayuda en el desarrollo de este Seminario, por facilitarnos los equipos necesarios para el avance de este proyecto.

También al profesor Cristian Vidal, por la ayuda entregada al comienzo de esta investigación.

Agradecer también al profesor Marcelo Trujillo, por su disposición, por encaminarnos en nuestro tema y también por todo el tiempo que nos ha brindado en el desarrollo de este seminario.

Y como no agradecer a la Sra. Verónica Valenzuela por la amabilidad y ayuda prestada al facilitarnos el laboratorio en donde trabajamos.

SUMARIO

Esta investigación tiene por objetivo, como su nombre bien lo indica, realizar la implementación de un servidor Web sobre un Cluster corriendo en Linux.

Este es un estudio de tipo exploratorio, ya que tiene por objeto esencial mostrar la implementación del mencionado Cluster. En ese sentido, pretende ser un punto de partida para estudios posteriores de mayor profundidad.

En esta investigación se estudiaron los distintos tipos Cluster, sus características y sus aplicaciones, para luego seleccionar aquellas implementaciones que cumplían de mejor forma los objetivos planteados.

Dentro de las implementaciones realizadas, se encuentran un Cluster de Alta-Disponibilidad (UltraMonkey) y un Cluster Alto-Rendimiento (openMosix).

Dentro de este trabajo se podrá encontrar que de las soluciones investigadas, la más óptima para la problemática planteada es la ofrecida por UltraMonkey, ya que cumple con todos los requisitos que se han puesto en esta investigación, mientras que con openMosix nunca fue posible solucionar el problema de memoria compartida.

El Cluster UltraMonkey es viable para ser implementado en cualquier pequeña y mediana empresa (Pymes) ya que, por ser de código abierto, no representa para estas organizaciones ningún costo por concepto de licencias.

ÍNDICE GENERAL

Apartado Nº 3

Apartado Nº 4

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

Apartado N° 1
Introducción

1.0 INTRODUCCIÓN

Este proyecto nace por una inquietud originada en la Dirección De Informática (DDI) de la Universidad Católica del Maule. Esta inquietud consistía en conseguir la implementación de un Cluster que balanceara la carga de distintos servidores Web, y de esta forma, prestar un servicio mucho más ágil, rápido y sin interrupciones en el usuario.

Se debe entender por Cluster a "un conjunto de máquinas unidas por una red de comunicación trabajando por un servicio conjunto" [CAT2004].

Los costos económicos para contar con una gran computadora (Mainframes) que entregue un servicio como el que se necesita en este caso, son bastantes elevados si se les compara con los costos que implica la implementación de Cluster.

Como los costos del equipamiento no son tan elevados para realizar la implementación de un Cluster que preste el servicio requerido, se puede considerar como una solución viable, para ser utilizado en las pequeñas y medianas empresas (Pymes).

Cabe mencionar que la plataforma que se utilizará para realizar esta implementación, como claramente lo señala el título de este seminario, es Linux.

Otro punto de este proyecto está dado por la utilización solamente de software de código abierto, y por lo tanto en la organización que se implemente esta alternativa, no se tendrá que incurrir en gastos de licencias.

Dos conceptos importantes de entender antes de mencionar los objetivos son: paralelismo y balanceo de carga.

"El paralelismo consiste en poder dividir una tarea en partes que trabajen independientemente en lugar de poseer una única tarea en la que todos sus procesos se encadenan uno tras otro, necesitando de los resultados del anterior para comenzar" [BRO2004].

"El balance o balanceo de carga es un concepto usado en informática que se refiere a la técnica usada para compartir el trabajo a realizar entre varios procesos, ordenadores, discos u otros recursos. Está íntimamente ligado a los sistemas de multiprocesamiento, o que hacen uso de más de una unidad de procesamiento para realizar labores útiles" [WIK2004b].

1.1 OBJETIVOS

Objetivo General.

Implementar un Cluster que permita el balanceo de carga entre los diferentes servidores WEB Apache.

Objetivos Específicos.

- Investigar las diferentes clases de Cluster existentes y seleccionar las soluciones viables.
- Implementar distintos tipos de Cluster, en plataforma Linux.
- Seleccionar entre los distintos tipos de Cluster, con respecto al servicio Web que se quiere implementar.
- Obtener una posible solución para ser implementada en Pymes.

Para desarrollar este proyecto se han investigado tanto los distintos tipos de Cluster como el software que permite la implementación de los mismos; dentro del software se encuentra el proyecto openMosix, con todas sus características. Este software forma parte de los denominados Cluster de Alto-Rendimiento. También se encuentra el proyecto LVS (Linux Virtual Server) el cual forma parte de los Cluster de Alta-Disponibilidad.

A través de un proceso de selección se obtendrá una alternativa que de solución a los distintos objetivos planteados.

Las restricciones de este proyecto es que debe ser implementado en Linux, y montar sobre el Cluster un servidor Web Apache. Considerando que estos son de licencias GPL.

1.2 ORGANIZACION

La estructura de este seminario está organizado de la siguiente manera:

En el Apartado Nº1 se realiza la introducción a este seminario, considerando tanto los objetivos, como la explicación de los distintos apartados.

El Apartado Nº2 trata del marco teórico que compone esta memoria, explicando lo que son los Cluster, sus principales características y las distintas clasificaciones que se pueden realizar de éstos.

También en este apartado se tratan los Cluster de Alta-Disponibilidad, sus características, los métodos de balanceo de carga, y la solución UltraMonkey, que es la que mejor cumple con los objetivos planteados en esta investigación.

Otro punto importante de este apartado son los Cluster de Alto-Rendimiento, siendo el principal tema openMosix, sus características, las técnicas de monitorización y el problema de la memoria compartida.

El Apartado N°3 nos explica como llevar a cabo las implementaciones, partiendo por la implementación de UltraMonkey, y luego se explica como se debe realizar la implementación de openMosix.

El Apartado N°4 nos muestra las distintas pruebas aplicadas a las implementaciones realizadas.

Por último se encuentran las Conclusiones de este seminario, las referencias bibliográficas y el Glosario.

Apartado N° 2
Marco Teórico

2.0 MARCO TEÓRICO

En este apartado se desarrollará todo el marco teórico necesario para entender este seminario. En él encontramos los siguientes temas:

- Cluster, lo que son, sus principales características, sus clasificaciones, etc.
- Cluster de Alta-Disponibilidad, utilizando la solución Linux Virtual Server (LVS).
- Cluster de Alto-Rendimiento, siendo utilizada la solución openMosix.

2.1 CLUSTER

En esta sección se tratará el tema de Cluster, ya que éste es el tema principal de este seminario, se tratarán los siguientes puntos:

- Que son los Cluster.
- Sus principales características.
- Factores de diseño para clasificar un Cluster.
- Las distintas clasificaciones de los Cluster.

2.1.1 ¿QUÉ ES UN CLUSTER?

Existe un gran inconveniente para definir lo que es, en si, un Cluster de computadoras, dado que no se ha llegado a un consenso de lo que realmente es.

Una buena definición de lo que es un Cluster, es la que señala [CAT2004], que dice: "Un conjunto de máquinas unidas por una red de comunicación trabajando por un servicio conjunto". Se debe entender por máquinas, a computadoras personales y no a cualquier tipo de máquina conectada a una red (Ej. Impresoras, consolas de juegos, etc.). La idea es que este

tipo de máquinas estén dotadas de lo primordial para llevar a cabo un proceso, es decir, de procesador y de memoria.

Otra forma de entender un Cluster está dada por el autor señalado anteriormente, que lo menciona como “una variación de bajo precio de un multiprocesador masivamente paralelo (miles de procesadores, memoria distribuida, red de baja latencia), con las siguientes diferencias: cada nodo es un máquina, quizás sin algo de hardware (monitor, teclado, mouse, etc.)”

Cabe mencionar que la unidad básica de un Cluster se llama “nodo”, el cual es un computador personal; estos nodos deben estar comunicados por un canal de conexión. Para esta investigación fue utilizada una red Ethernet.

Para dejar más claro lo que es un Cluster, veremos un ejemplo mostrado en la Figura Nº 2.1 “Ejemplo de máquinas para implementar un Cluster”.

Uno de los factores de gran importancia, cuando se habla de conseguir la eficiencia del Cluster, es el ancho de banda con el que se dispone para establecer la comunicación entre los nodos, dado que a mayor ancho de banda, más rápido se transferirán los diferentes paquetes, consiguiendo así, reducir el tiempo de latencia.

Figura N° 2.1 Ejemplo de máquinas para implementar un Cluster [CHI2003].

Otro factor relevante, que no se puede dejar de mencionar, es el sistema operativo en el cual es implementado el Cluster, ya que de éste depende como se administran los recursos de hardware, y a que nivel se realicen las transferencias de los paquetes de información, entre otros factores.

2.1.2- CARACTERÍSTICAS DE UN CLUSTER

Si se habla de Cluster y de los tipos de Cluster existentes, es necesario mencionar cuales son las cualidades comunes que presentan. Algunas de ellas ya han sido nombradas anteriormente, pero ahora se describirán de una manera más formal. Para ello, se mencionarán las características de los Cluster definidas, según [HER2005]:

- **Un Cluster consta de 2 o más nodos.** Un sólo computador, en ningún caso, puede ser considerado como un Cluster debido a la situación de aislamiento en que se encuentra, puesto que no puede comunicarse y menos, ocupar los recursos de otra máquina.

- **Los nodos de un Cluster deben estar conectados entre si por, al menos, un canal de comunicación.** De no ser así, se produce el efecto de aislamiento anteriormente mencionado.

- **Los Cluster necesitan software de control especializado**. Se debe tener presente que el software utilizado es el que determinará el tipo de Cluster que se está implementando. Además, parte de este software es el encargado de la comunicación entre los componentes del Cluster. El software utilizado puede ser de uno de los siguiente niveles:

    - Software a Nivel de aplicación. Para la utilización de este software se emplean librerías, las cuales son de carácter general y permiten el

comportamiento del Cluster como un solo gran sistema. Se puede ver graficado en la parte inferior de la Figura Nº2.2.

- Software a Nivel de Sistema. Este tipo de software puede ser una parte del operativo o la totalidad de éste. Este nivel es más complejo, pero la eficiencia que brinda, por norma general, es superior a los de nivel de aplicación. Se puede ver graficado en la parte superior de la Figura Nº2.2.

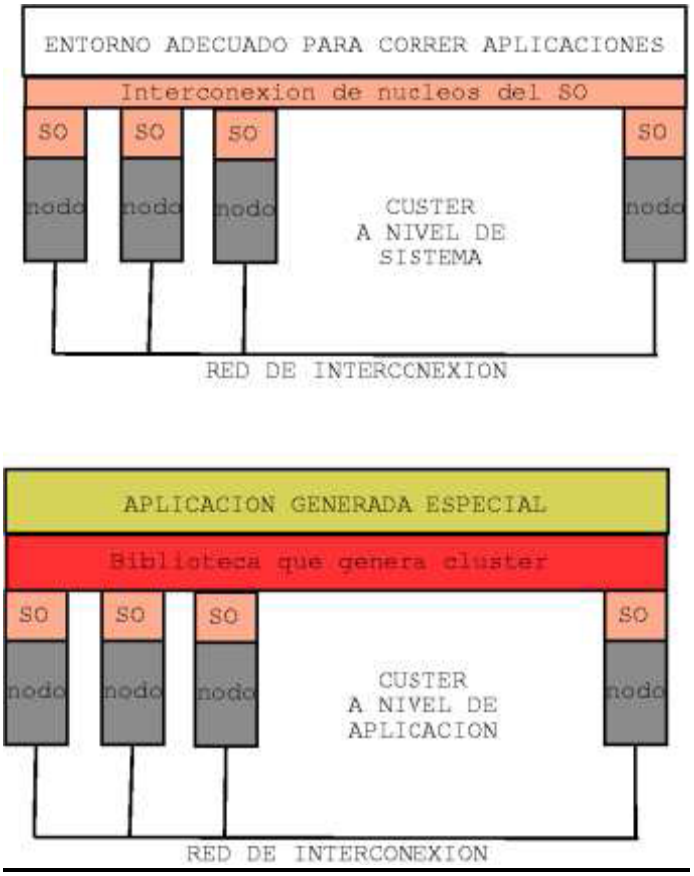

Figura Nº2.2 Niveles del Software.

2.1.3- FACTORES DE DISEÑO PARA CLASIFICAR UN CLUSTER

Existen diferentes factores de diseños según los cuales se puede clasificar un Cluster. Entre éstos encontramos los siguientes:

- Acoplamiento.
- Control.
- Homogeneidad.
- Escalabilidad.

2.1.3.1- Acoplamiento

Una de las características más importantes de un Cluster es el nivel de acoplamiento del mismo.

Por acoplamiento del software se entiende la integración que tengan los elementos existentes en cada nodo [CAT2004].

Los distintos tipos de acoplamiento son los que se describen a continuación:

- **Escasamente acoplados** — Una agrupación de computadores está escasamente acoplada si, aún siendo capaz de realizar procesamiento paralelo mediante librerías de paso de mensajes o de memoria compartida, no posee un sistema de instalación y gestión integrado que posibilite una recuperación rápida ante fallos y una gestión centralizada que ahorre tiempo al administrador [RID2003].

- **Medianamente acoplados** — Dentro de este grupo se encuentra un software que no necesita un conocimiento tan profundo sobre cuales son los recursos de los otros nodos que componen el Cluster, pero utiliza el software de otros nodos para realizar aplicaciones de muy bajo nivel. Un ejemplo de este tipo de acoplamiento es

openMosix y Linux-HA (Alta-Disponibilidad). Una nota importante, es que un Cluster openMosix necesita que todos los Kernels sean de la misma versión [CAT2004].

- **Altamente acoplados** — Este software se caracteriza por que los elementos que lo componen se interrelacionan unos con otros y posibilitan la mayoría de las funcionalidades del Cluster de manera altamente cooperativa. El acoplamiento más fuerte que se puede dar se produce cuando existe sólo una imagen del sistema operativo, la cual está distribuida entre el conjunto de nodos que la compartirán.
  Este caso es el que se considera como más acoplado, de hecho, no está catalogado como Cluster, sino como sistema operativo distribuido.

2.1.3.2- Control

Cuando se habla de control de un Cluster, no es más que el modelo de gestión que éste propone. Estos modelos pueden ser de dos tipos, éstos son:

- **Control centralizado**: En este tipo de control existe un nodo maestro desde el cual se realiza la configuración de todo el Cluster. Además ayuda a que la gestión y la administración sean mucho más fácil de realizar, pero a su vez los hace menos tolerable a los fallos.

- **Control descentralizado:** En este tipo de control cada uno de los nodos del Cluster debe ser capaz de administrarse y gestionarse. En este tipo de control se hace más difícil la gestión y la administración, pero como sistema global lo hace más tolerable a fallos.

2.1.3.3- Homogeneidad

Se entiende por homogeneidad de un Cluster a lo similar que pueden llegar a ser los equipos y recursos que conforman éste. Se clasifican en:

- **Cluster homogéneos:** En este tipo de Cluster todos los nodos que lo componen poseen arquitectura y recursos similares, es decir, no debe existir mucha diferencia entre cada nodo.

- **Cluster heterogéneos:** Este tipo de Cluster está formado con nodos en los cuales pueden existir las siguientes diferencias:

    - Tiempos de acceso.
    - Arquitectura.
    - Sistema operativo.
    - Rendimiento de los procesadores o recursos sobre una misma arquitectura.

  El uso de arquitecturas distintas o distintos sistemas operativos, impone que exista una biblioteca que haga de interfaz [CAT2004].

2.1.3.4.- Escalabilidad

Otro factor de suma importancia que aún no se ha nombrado es el de "Escalabilidad del Cluster".

Escalabilidad es la capacidad de un sistema informático de adaptarse a un número de usuarios cada vez mayor, sin perder calidad en los servicios. En general, se podría definir como la capacidad del sistema informático de cambiar su tamaño o configuración para adaptarse a las circunstancias cambiantes [WIK2006b].

Por lo tanto, entre más escalable es un sistema, menos costará mejorar el rendimiento, lo cual abarata el coste y, en caso de que un Cluster lo implemente, distribuye más la caída del sistema [CAT2004].

2.1.4.- CLASIFICACIÓN DE CLUSTERS

La forma en que operará el Cluster está determinada por la función que éste deberá desempeñar.

De esta manera, la forma de operar es a su vez definida por el software de control especializado que se detalló anteriormente. Según [CAT2004] los distintos Cluster existentes son los siguientes:

- Cluster de Alto-Rendimiento
- Cluster de Alta-Disponibilidad
- Cluster de Alta-Confiabilidad.

2.1.4.1- Cluster de Alto-Rendimiento (HP, High Performance)

Un **Cluster de Alto-Rendimiento** es aquel que está diseñado para dar altas prestaciones en cuanto a capacidad de cálculo [WIK2006c].

De acuerdo con la definición, el recurso que comparten los nodos es el más importante de una máquina, éste es, el tiempo de proceso.

El objetivo de este tipo de Clusters es, como su propio nombre indica, mejorar el rendimiento en la obtención de la solución de un problema, en términos bien del tiempo de respuesta, bien de su precisión [CAT2004].

Existen distintas aplicaciones que se les puede dar a este tipo de Cluster, entre las cuales encontramos las siguientes:

- Cálculos matemáticos.
- Renderizaciones de gráficos.

- Compilación de programas.
- Compresión de datos.
- Descifrado de códigos.
- Rendimiento del sistema operativo, (incluyendo en él, el rendimiento de los recursos de cada nodo).

Por lo tanto se puede decir que este tipo de Cluster puede ser utilizado en problemas que requieran grandes tiempos de proceso, siempre y cuando se encuentre un algoritmo paralelizable.

Existen Clusters que pueden ser denominados de Alto-Rendimiento tanto a nivel de sistema como a nivel de aplicación. A nivel de sistema existe openMosix, mientras que a nivel de aplicación se encuentran algunos como MPI, PVM, Beowulf y muchos otros.

En cualquier caso, estos Clusters hacen uso de la capacidad de procesamiento que pueden tener varias máquinas [CAT2004].

Las implementaciones a nivel de aplicación no suelen implementar balanceo de carga, a diferencia de las implementaciones a nivel de Kernel que si lo hacen, además de compartir los recursos a cualquier nivel.

Los aspectos de implementación para este tipo de los Cluster son las siguientes:

2.1.4.1.1- Asignación de procesos a los nodos

Los procesos pueden ser ejecutados en cualquier nodo del Cluster, pero donde permanecerá el resto de su vida está dado por las siguientes asignaciones:

- Estática: Se elige estáticamente el nodo donde el proceso vivirá toda su vida [CAT2004]. Al ser estático puede producir un mal balanceo de la carga.

- Dinámica: Los procesos una vez iniciados en un nodo pueden migrar a otro nodo dinámicamente [CAT2004]. En este caso el algoritmo que realiza la migración debe tener mucho cuidado con el balanceo de la carga, y sobre qué variables está tomando las decisiones de migración. Por lo tanto su implementación es compleja, y también se pueden producir una sobrecarga en el Cluster.

2.1.4.1.2- Requisa

Se entiende por Requisa al hecho de poder parar un determinado proceso y poder adquirir sus recursos. Por lo tanto los Cluster de Alto-Rendimiento pueden implementar o no la Requisa.

La ventaja de implementar Requisa consiste en que el tiempo de latencia en los procesos de mayor prioridad será menor, pero la desventaja es que se pueden producir sobrecargas en el Cluster y que la complejidad al realizar la implementación aumente.

2.1.4.1.3- Modos de dedicar los Nodos

Existen modos de dedicar los nodos para llevar a cabo un determinado proceso, los modos de dedicar los nodos son los siguientes:

- Modo dedicado: En este modo, que es el más simple de todos, solamente un trabajo está siendo ejecutado en el Cluster en un tiempo dado, y como mucho un proceso de este trabajo que se está ejecutando es asignado a un nodo en cualquier momento en el que se siga ejecutando el trabajo. Este trabajo no liberará el Cluster hasta que acabe completamente aunque solamente quede un proceso ejecutándose en un único nodo. Todos los recursos se dedican a este trabajo. Como se puede comprender fácilmente, esta forma de uso de un Cluster puede llevar a una pérdida importante de potencia sobre todo si no todos los nodos acaban el trabajo al mismo tiempo [CAT2004].

- Modo de división en el espacio: En este modo existen particiones disjuntas del Cluster, en las cuales se ejecuta sólo un trabajo en la partición dada, parecido al modo anterior, lo que puede ocasionar algunos problemas como que la partición no sea lo suficientemente grande como para poder llevar a cabo el trabajo o que sea muy pequeño el trabajo y se desperdicien gran cantidad de nodos de la partición.

- Modo de división en el tiempo: En cada nodo pueden estar ejecutándose varios procesos a la vez por lo que se solucionan los problemas anteriores. Este es el modo más usado normalmente puesto que no tiene tantas restricciones como los anteriores y se puede intentar hacer un equilibrio de carga eligiendo correctamente los procesos.

2.1.4.2- Cluster de Alta-Disponibilidad (HA, High Availability):

Este tipo de Cluster es totalmente diferente a los Cluster de Alto-Rendimiento. Por el contrario los Clusters de Alta-Disponibilidad están diseñados para garantizar el funcionamiento ininterrumpido de ciertas aplicaciones [GAR2003].

La idea principal de este tipo de Cluster es proporcionar un servicio ininterrumpido las 24 horas del día, los 7 días de la semana.

Un Cluster de Alta-Disponibilidad es un conjunto de dos o más máquinas, que se caracterizan porque comparten los discos de almacenamiento de datos, y porque están constantemente monitorizándose entre si. Si se produce un fallo del hardware o de las aplicaciones de alguna de las máquinas del Cluster, el software de Alta-Disponibilidad es capaz de rearrancar automáticamente los servicios que han fallado en cualquiera de las otras máquinas del Cluster. Y cuando la máquina que ha fallado se recupera, los servicios son nuevamente migrados a la máquina original. Esta capacidad de recuperación automática de servicios garantiza la integridad de la información, ya que no hay pérdida de datos, y además evita molestias a los usuarios, que no tienen por qué notar que se ha producido un problema [WIK2005].

Para poder cumplir con el objetivo de ser capaz de estar siempre dando servicios, este tipo de Cluster se implementa en base a tres factores, los cuales son:

- Fiabilidad: Probabilidad de un funcionamiento correcto [OLE2004].
- Disponibilidad: La calidad de estar siempre presente, listo para el uso, a mano, accesible [OLE2004].
- Dotación de servicio: Debe existir un servicio proporcionado por el Cluster.

Este tipo de Cluster está diseñado para resolver múltiples problemas dentro de los cuales encontramos los siguientes:

- Sistemas de información redundante.
- Sistemas tolerantes a fallos.
- Balanceo de carga entre varios servidores.
- Balanceo de conexiones entre varios servidores.

Como se puede apreciar las necesidades que se pretenden solucionar son: tener un servicio disponible y ahorrar económicamente todo lo que sea posible [CAT2004].

2.1.4.3- Cluster de Alta-Confiabilidad (HR, High Reliability)

Estos tipos de Clusters son los más difíciles de implementar. No se basan solamente en conceder servicios de Alta-Disponibilidad, sino en ofrecer un entorno de sistema altamente confiable. Esto implica muchísima sobrecarga en el sistema, son también Clusters muy acoplados [CAT2004].

Dar a un cluster SSI capacidad de Alta-Confiabilidad implica gastar recursos necesarios para evitar que aplicaciones caigan [CAT2004].

La idea de este tipo de Cluster es que cuando un servicio se ha caído, éste sea relanzado utilizando el último checkpoint (o punto de parada) del servidor anterior, además otra de sus funciones es mantener el estado de las aplicaciones.

Generalmente este tipo de Clusters suele ser utilizado para entornos de tipo empresarial y esta funcionalidad solamente puede ser efectuada por hardware especializado. Por el momento no existe ninguno de estos Clusters implementados como software. Esto se debe a limitaciones de la latencia de la red, así como a la complejidad de mantener los estados [CAT2004].

2.2.- CLUSTER DE ALTA-DISPONIBILIDAD: LINUX VIRTUAL SERVER

Linux Virtual Server (**LVS**) es una solución para gestionar balance de carga en sistemas Linux. Es un proyecto de código abierto iniciado por Wensong Zhang en mayo de 1998. El objetivo es desarrollar un servidor Linux de alto rendimiento que proporcione buena escalabilidad, confiabilidad y robustez usando tecnología Clustering [WIK2006].

El Linux Virtual Server es un servidor altamente escalable y altamente disponible construido sobre un Cluster de servidores reales, con el balanceador de carga corriendo en un sistema operativo Linux. La arquitectura del Cluster servidor es completamente transparente a los usuarios finales, y los usuarios interactúan como si fuera solamente un solo servidor virtual de Alto-Rendimiento [LVS2005]. La figura Nº2.3 “Linux Virtual Server” ilustra el funcionamiento de un LVS.

La principal idea es proveer de un mecanismo de migración de sockets. El mecanismo se basa en utilizar una máquina servidora a la que se dirigen las peticiones de los usuarios clientes. La interfaz pública (en Internet) de esta máquina normalmente tiene asociada una dirección conocida como VIP. El cometido de esta primera computadora es direccionar dichas peticiones a otros servidores reales mediante varias técnicas, de este modo los usuarios clientes ven un único servidor. No obstante éste opera con varias máquinas para conceder un servicio único al exterior [CAT2004a].

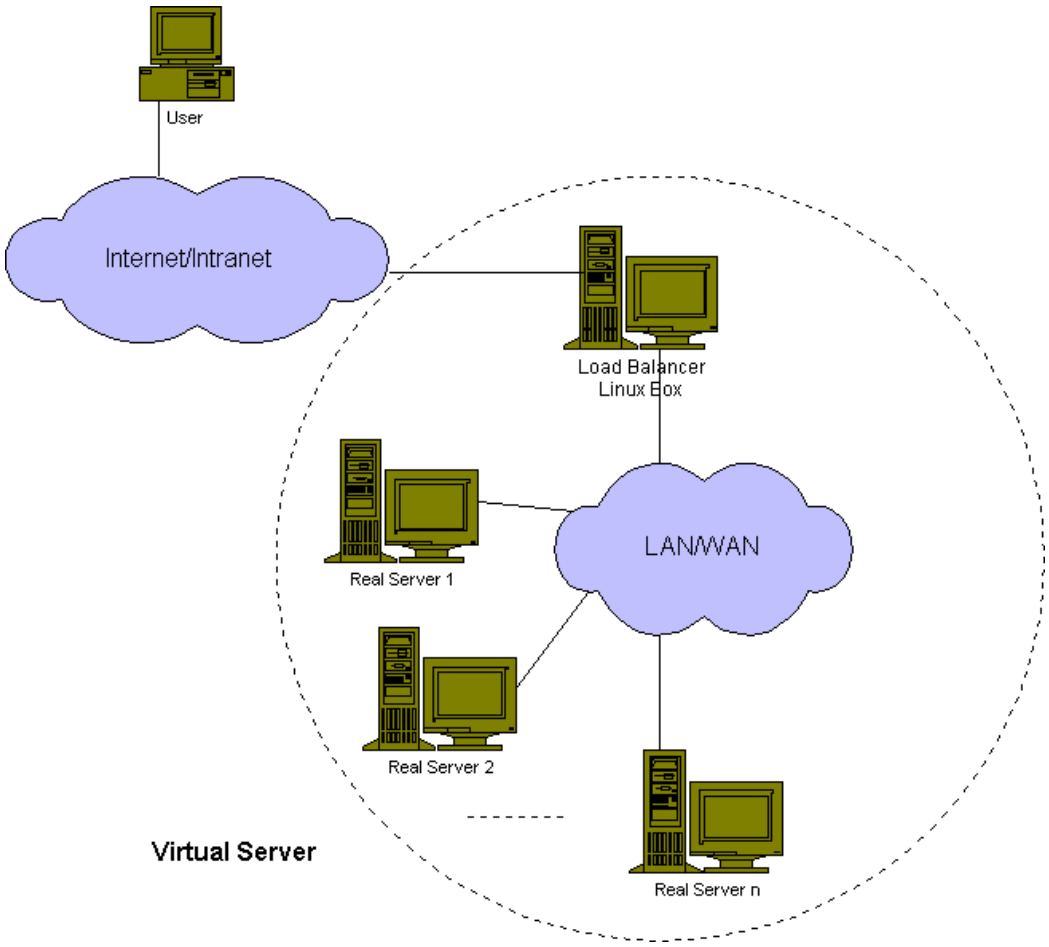

Figura N°2.3 Linux Virtual Sever [LVS2005].

2.2.1- MÉTODOS DE BALANCEO IP

En este tipo de Cluster el nodo director actúa como una especie de router. En el Kernel del nodo se encuentran añadidas todas las tablas de encaminamiento, las cuales son empleadas para realizar el reenvío de los paquetes a los servidores reales. Para efectuar este reenvío de paquetes existen tres formas las cuales son:

- VS-NAT
- VS-DR
- VS-TUN.

2.2.1.1- VS-NAT

Es el caso más sencillo de configurar de todos y el que menor rendimiento tiene respecto a los otros dos. VS-NAT hace uso de NAT para modificar direcciones. Existen tanto la implementación para las versiones de Kernel 2.25 como para las 2.46. Ambas implementaciones dan soporte SMP para LVS en el nodo director (que es el que tiene el Kernel modificado), lo que permite una tasa de manejo de paquetes muy alta para Clusters que proveen de mucho servicio [CAT2004a].

Para que este tipo de método de balanceo funcione de manera adecuada, se debe contar con el Kernel del director parchado con LVS (ipvs) y a su vez de una batería de servidores, los cuales pueden correr cualquier sistema operativo y cualquier servicio.

La tarea realizada por el nodo director es recibir las peticiones de los clientes por su VIP, y éste a su vez reenvía los paquetes al servidor real, el cual responde a la petición y los envía nuevamente al nodo director, el que cambia las direcciones de cabecera para que no existan problemas a la hora en que el cliente reciba dichos paquetes.

Como se puede ver, el mecanismo es muy parecido, por no decir igual, que el de un proxy inverso, excepto por que el redireccionamiento se hace a nivel de Kernel.

Una ilustración clara de este tipo de método de balanceo es el mostrado en la figura N°2.4 “VS-NAT”.

VS-NAT tiene el mismo problema que los proxys inversos: el nodo director llega a ser un cuello de botella en cuanto las exigencias por parte de los clientes se hacen muy altas, o el número de servidores internos de la red crece por encima de los 20. Es por esto que este tipo de configuración es la menos utilizada de las tres [CAT2004a].

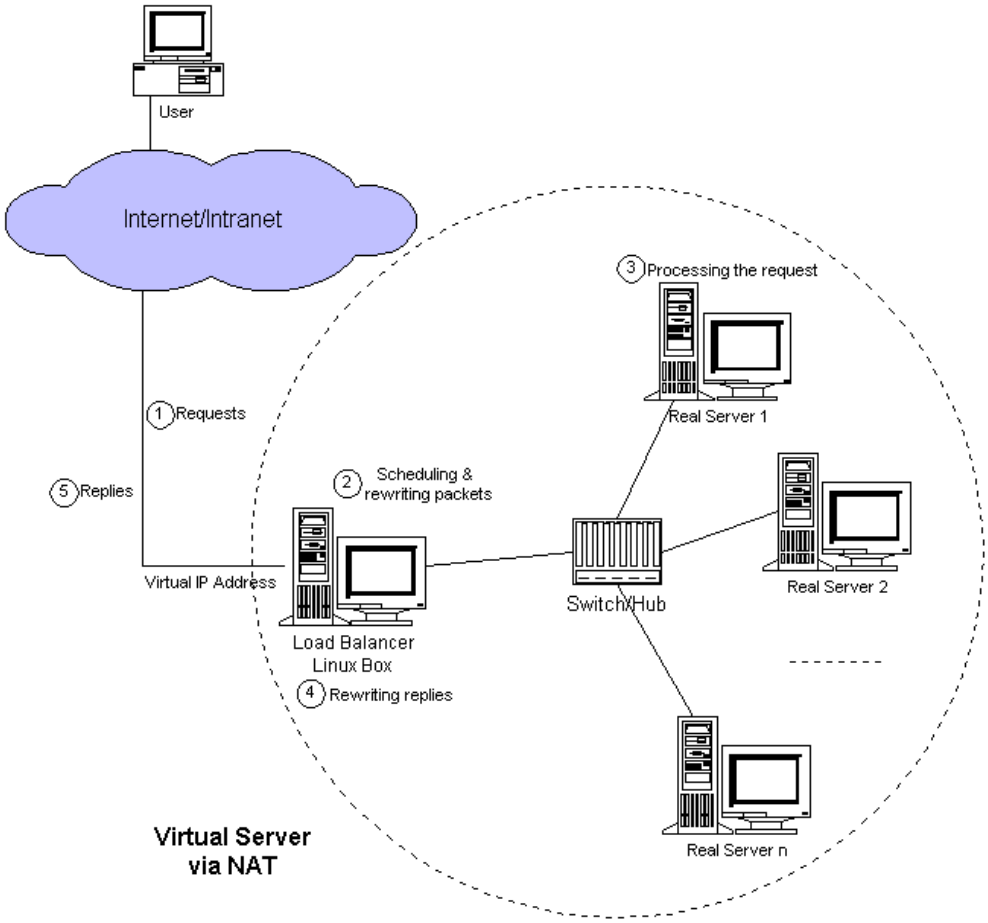

Figura N°2.4 VS-NAT [LVS2005].

2.2.1.2- VS-TUN

Este método es más utilizado que el anterior, se basa en redirigir los paquetes IP del nodo director al destino mediante técnicas de IP-tunneling. Esto requiere que tanto el nodo director (que debe correr bajo Linux y por tanto puede ser compilado con ***IP-tunneling***) como el servidor real puedan encapsular y desencapsular paquetes especiales. Para esto es necesario que la pila IP del sistema operativo lo soporte, y no todos los sistemas operativos lo soportan. En general la mayoría de Unix que existen en el mercado si lo soportan, por lo que en un principio no debe ser un grave inconveniente para la elección de este método como base de LVS [CAT2004a].

El funcionamiento mediante este método de balanceo es el siguiente:

- El cliente hace la petición a la VIP del director.
- El director elige mediante el algoritmo de balanceo cuál será el servidor real que atienda la petición.
- El servidor encapsula el paquete (que le llegó por la interfaz asignada a la VIP) en otro paquete IP con destino al servidor real.

La función que cumple el servidor real es atender las peticiones de servicio y responder, utilizando la dirección con la cual llegó el servicio, es decir, la VIP. Un aspecto importante es que el servidor real envía directamente los paquetes al cliente sin tener necesidad de volver a pasar por el nodo director. De esta manera se evita el problema de cuello de botella en el director, como ocurre en VS-NAT.

Otra característica importante en este tipo de redireccionamiento está dada por que se puede estar trabajando en una red WAN, por lo tanto los servidores reales pueden estar dispersos por cualquier parte del mundo, a las cuales se puede acceder desde el director. Esto nos ayuda a no sólo distribuir los servicios, sino que también poder realizar una distribución geográfica de los servidores.

Una ilustración clara de este tipo de método de balanceo es el mostrado en la figura N°2.5 “VS-TUN”.

Una de las desventajas que tiene este tipo de técnica es que tanto el director como el servidor tienen que ser capaces de crear interfaces de tipo tunneling, y como consecuencia de hacer IP-tunneling siempre estaría implícito un tiempo de procesador ocupado en encapsular o desencapsular los paquetes, que si bien en algunos sistemas puede ser insignificantes, en otros puede ser decisivo para la elección de otro método de balanceo.

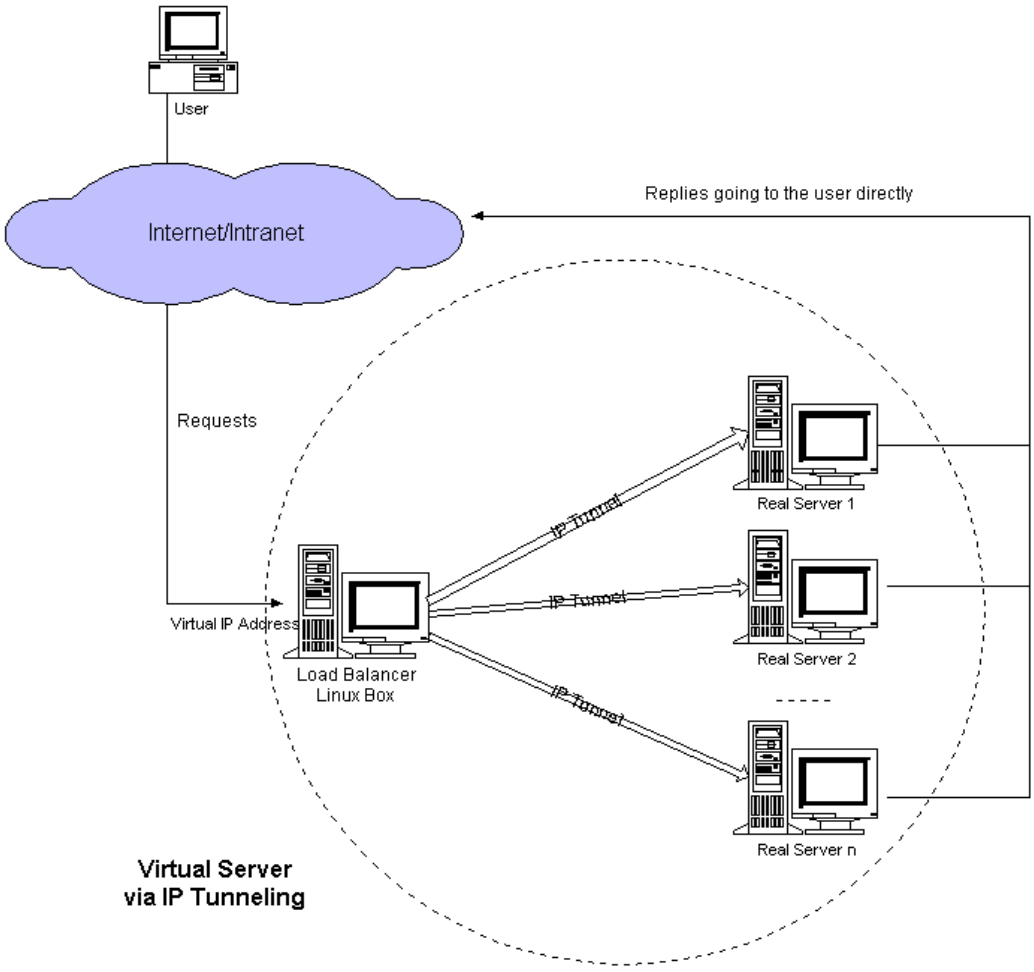

Figura Nº2.5 VS-TUN [LVS2005].

2.2.1.3.- VS-DR

Es una tecnología que realiza un encaminamiento directo de los paquetes del nodo director al los servidores reales. Su base en una tecnología de red local (en un único segmento) y en un cambio de direcciones IP-MAC proporcionan el método de reenvío de los paquetes.

Una similitud que se presenta con VS-TUN es que ambos métodos no reenvían los paquetes al nodo director, de esta forma, tampoco existe el cuello de botella presentado en VS-NAT. Por ser el que mayor rendimiento obtiene, es uno de los más utilizados, pero al igual que el resto, presenta una serie de desventajas en su uso y configuración.

El funcionamiento de VS-DR es similar al de VS-TUN en el sentido de que ambos utilizan la dirección VIP no solamente en el nodo director (donde está la dirección VIP real a la que acceden los clientes) sino también en los nodos servidores. En este caso, los servidores poseen dos direcciones asociadas al nodo, una es la IP real asociada a la tarjeta Ethernet, la otra es una dirección loopback especial configurada con la dirección VIP, es conveniente dejar la interfaz loopback que tiene la dirección 127.0.0.1 sin modificar, por lo cual se debe hacer un alias de esta interfaz pero con la dirección conocida como VIP [LVS2005].

Una ilustración clara de este método de balanceo es el mostrado en la Figura N°2.6 “VS-DR”.

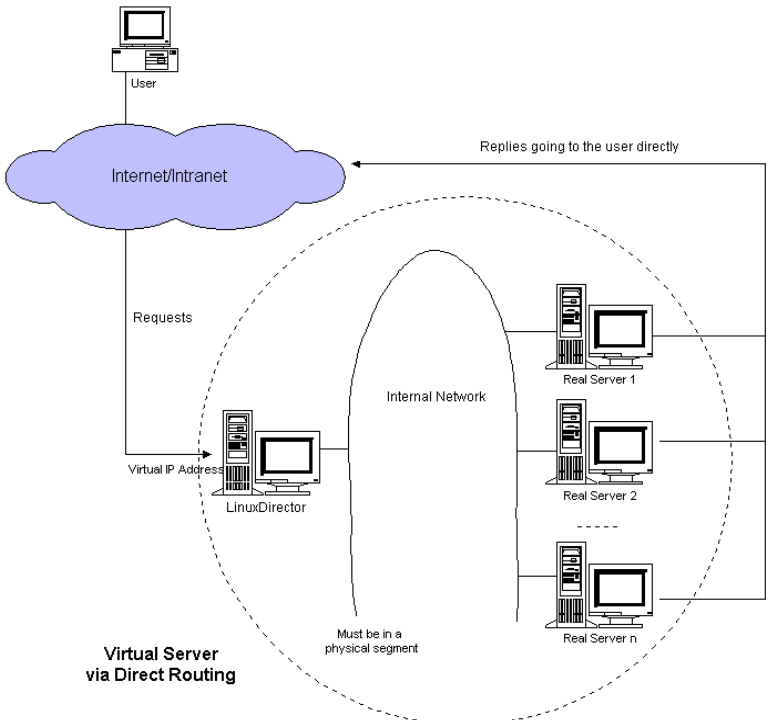

Figura N°2.6 VS-DR [LVS2005]

Cuando un cliente quiere realizar una conexión con el director lanza un paquete ARP (o en su defecto lo manda el router, en cualquier caso el problema es el mismo) para saber cuál es la dirección MAC del director. Éste hace el reenvío dependiendo del algoritmo de planificación y al mismo tiempo guarda en su tabla el estado de la conexión que se ha realizado. El servidor contesta y se reenvía la respuesta hacia el cliente [CAT2004a].

Puede darse el caso que no ocurra así y en vez de contestar el director conteste uno de los servidores reales, por lo cual la conexión se realizará pasando por encima del LVS, en una especie de by-pass, y de esta forma el nodo director no guardará ni los estados ni la conexión. Hasta aquí no existe ningún problema.

El problema está en el caso de que la tabla ARP del router o cliente se borre y vuelva a pedir ARP Who has VIP tell cliente/router concediéndose la MAC de otro servidor o incluso del nodo director, que no tiene ni idea de cómo va la conexión en ese momento ya que el no estaba entregando el servicio Figura Nº2.7 “Problema ARP”.

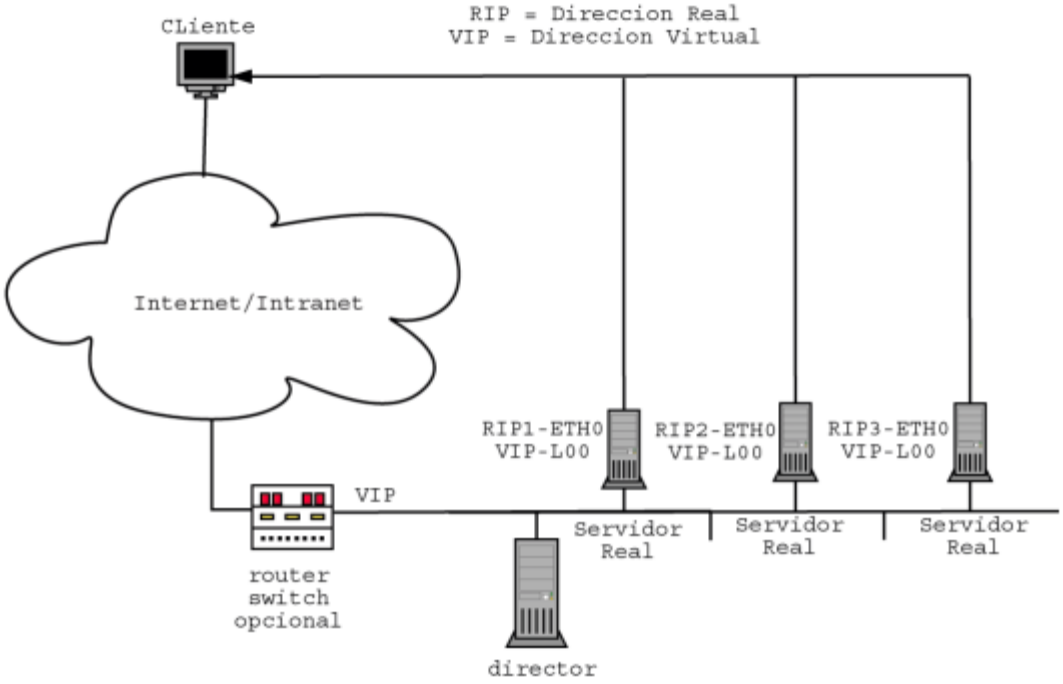

Figura N°2.7 Problema ARP [WIK2004].

Como se puede ver es un problema inherente a la manera en la que se ha configurado la solución, la que podría ser resuelta poniendo más de una tarjeta de red al director entre otras soluciones. De cualquier forma, la red se comporta normalmente de manera aleatoria en la concesión de la MAC.

Existen varias soluciones, desde parches para el Kernel hasta un comando ifconfig -arp, pasando por meter una tarjeta de red más en el director o incluso modificando la tabla ARP del router para que no solicite ARP dinámicamente sino lo especificado.

2.2.2 LA SOLUCIÓN ULTRAMONKEY

UltraMonkey es un proyecto que busca conseguir balanceo de carga y Alta-Disponibilidad en los servicios en una red de área local usando componentes de Fuente Abierta (Open Source) en un sistema operativo Linux, que incluye las herramientas Heartbeat y ldirectord del proyecto Linux-HA.

El servicio puede estar destinado a usuarios finales a través de todo el mundo, conectados vía Internet, o bien a usuarios de una empresa conectados vía intranet.

UltraMonkey hace uso del sistema operativo Linux porque busca proveer una solución flexible que pueda ser adaptada a un gran número de necesidades. De esta forma, se logra que de manera similar se puedan implementar desde Clusters pequeños de sólo dos nodos a grandes sistemas sirviendo miles de conexiones por segundo.

2.2.2.1.- Características de UltraMonkey

Las principales características de la solución UltraMonkey, son las que se detallan a continuación:

- Balanceo de Carga Rápido usando el Servidor Virtual Linux.
- Alta-Disponibilidad Flexible provista por el marco de Linux-HA.
- Monitoreo de niveles de servicio utilizando ldirectord.
- Soporte para topologías de Alta-Disponibilidad y/o Balanceo de Carga son ejemplos de configuración trabajados.

- Fácil expansibilidad a un gran número de IP basadas en servicios virtuales usando fwmarks paquetes pre-construidos para Debian Sarge y Red Hat Enterprise Linux 3, todo el código es fuente abierta.

A continuación se detallarán las características necesarias para la instalación de un Cluster de Alta-Capacidad, Alta-Disponibilidad y balanceo de carga utilizando UltraMonkey en Debian Sarge.

2.2.2.2.- Descripción de la Topología de la Red

Como muestra la Figura Nº2.8 “Topología de red UltraMonkey”, la red está compuesta por 4 computadores unidos mediante un router. Las direcciones IP 192.168.6.2 y 192.168.6.3 están asignadas a los nodos directores del Cluster y las direcciones IP 192.168.6.4 y 192.168.6.5 a los servidores reales. La VIP del Cluster es 192.168.6.240.

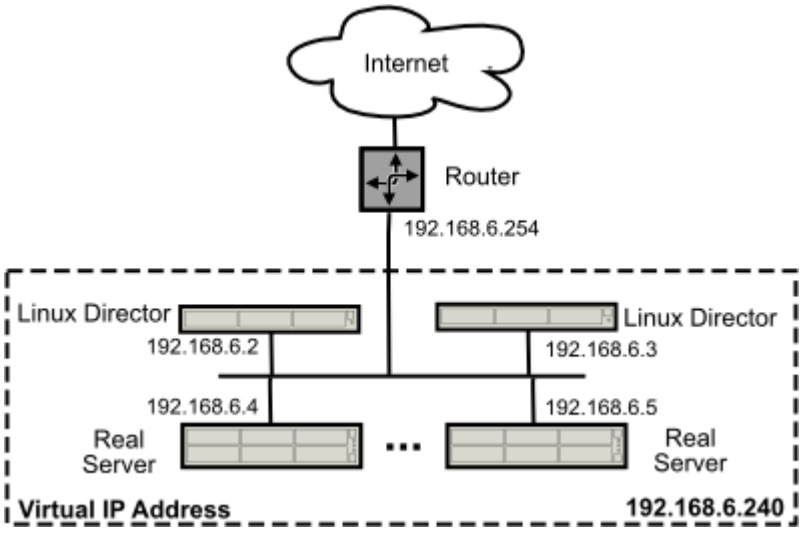

Figura N°2.8 Topología de red Ultramonkey [ULT2005].

Esta topología permite el máximo rendimiento (en cuanto a tasa de transferencia) a través de la red, ya que el tráfico de retorno ya no tiene que viajar a través de un nodo director.

2.2.2.3.- Directores-Linux

En cualquier momento uno de los nodos directores está activo, mientras el otro está en una espera atenta. El director Linux activo acepta el tráfico desde una IP virtual. Ésta es la IP que es anunciada a los usuarios finales a través de DNS. La carga de conexiones es balanceada hacia uno de los servidores reales usando LVS.

Los dos directores Linux se monitorean el uno al otro usando Heartbeat y en el evento de que el director Linux activo falle, el que está en espera asume la dirección virtual y el servicio se mantiene. Las conexiones son sincronizadas entre el nodo director activo y el(los) que está(n) en espera, de tal forma que cuando una falla ocurre las conexiones existentes pueden continuar.

Cuando un nodo director recibe una conexión desde un usuario final, éste toma la decisión acerca de a cuál servidor real enviar la conexión. Todos los paquetes del tiempo de vida de esta conexión son enviados al mismo servidor real de tal forma que la integridad de la conexión entre el usuario final y el servidor real se mantiene.

El director monitorea la salud de los servidores reales, por medio de consultas periódicas de una página conocida y chequeando que la respuesta contenga una cadena esperada. Si el servidor real falla, entonces el servidor es sacado del pool (los servidores reales seleccionables) de los servidores reales y se reinserta una vez que vuelve a estar en línea.

Como el director Linux no es el router de entrada para los servidores reales, el tráfico no tiene que viajar a través del director Linux en el retorno si es que se usa “ruteo directo” o “tunnelling”. Esto permite un mayor rendimiento (tasa de transferencia) ya que el director Linux no tiene que manejar los paquetes de retorno. Puede ser posible enviar tráfico de retorno a través de diferentes routers y/o switch cuando la conectividad externa lo permita.

2.2.2.4.- Servidores-Reales

Los Servidores Reales pueden ejecutar una variedad de servicios incluyendo el Servidor HTTP Apache. Más servidores reales pueden ser añadidos a la red si se requiere capacidad extra.

2.3.- CLUSTER DE ALTO RENDIMIENTO: OPENMOSIX

En esta sección se detallará toda la información relacionada con el Cluster de Alto-Rendimiento openMosix. Los temas a tratar serán:

- ¿Qué es openMosix?.
- Descripción de openMosix.
- Características.
- Técnicas de monitorización.
- Memoria Compartida.

2.3.1- ¿QUÉ ES OPENMOSIX?

El software openMosix permite realizar la implementación de un Cluster de Alto-Rendimiento, bajo el sistema operativo Linux. Una cualidad importante de openMosix es la posibilidad de realizar la implementación de un Cluster de Alto-Rendimiento, sin costos de licencias, esto es debido a que tiene licencia GPL2, lo cual es una de las características de esta investigación.

Este proyecto, openMosix, surge de la separación de los dos principales desarrolladores de Mosix, los cuales son Amnom Barak y Moshe Bar. Esta separación se produjo debido a diferencias de opinión sobre el futuro de la licencia de Mosix. Moshe Bar continuo desarrollando Mosix pero ahora llamado openMosix, la gran diferencia que existe es que este último es de licencia GPL2.

Es importante señalar que openMosix es un parche para el Kernel de Linux por lo que es compatible con el estándar de Linux para plataformas IA32. Los cambios se traducen entorno a un 3% del código total del Kernel, lo que realmente no es muy significativo.

2.3.2- DESCRIPCIÓN DE OPENMOSIX

El paquete de software openMosix convierte computadores corriendo GNU/LINUX conectados en red, en un Cluster. Éste realiza automáticamente el balanceo de carga entre los diferentes nodos del Cluster, y estos pueden unirse o retirarse del mismo sin presentar mayores problemas.

El software openMosix no dispone de un control centralizado y/o jerarquizado en maestros/esclavos: cada nodo puede operar como un sistema autónomo, y establece independientemente sus propias decisiones. Este diseño permite la configuración dinámica, donde los nodos pueden añadirse o dejar el Cluster con interrupciones mínimas [CAT2004].

También cabe mencionar que openMosix nos brinda una nueva dimensión de la escalabilidad de un Cluster bajo Linux. Permite la construcción de arquitecturas de alto rendimiento, donde la escalabilidad nunca se convierte en un cuello de botella para el rendimiento general del Cluster. Otra ventaja de los sistemas openMosix es la respuesta en condiciones de alta impredecibilidad producida por distintos usuarios y/o situaciones [CAT2004].

Los factores que son determinantes en el momento en que la carga es repartida entre los nodos son: su velocidad de conexión, el espacio libre con el que cuente la memoria y la velocidad de los procesadores en términos de ciclos por segundo.

La distribución de la carga de procesos es realizada por un algoritmo de balanceo de carga, el cual realiza en forma transparente (al usuario, al nodo y al proceso) las migraciones

de procesos de un nodo a otro. El beneficio es una mejor repartición de carga entre los nodos, ya que el mismo Cluster se preocupa de optimizar en todo momento las cargas de los nodos.

Además a través de las distintas herramientas de monitorización es posible visualizar cuál es la carga en cada equipo, y cómo el algoritmo de balanceo realiza la distribución.

También se ofrece la posibilidad de que el administrador del sistema pueda afectar el balance de carga automático mediante una configuración manual durante la ejecución de un determinado proceso.

Al momento de iniciar un proceso en un nodo "x", el Cluster determina, basado en diferentes criterios, si es mejor ejecutar un cierto proceso en un nodo menos cargado. El software openMosix utiliza un algoritmo avanzado basado en economía de mercado para determinar cuál nodo aloja mejor la aplicación. Por supuesto, el administrador del sistema puede siempre forzar a que una aplicación sea ejecutada en un cierto nodo. De esta forma, incluso las aplicaciones que han sido paralelizadas se benefician de openMosix.

El software openMosix es una de las más rápidas y fáciles vías para instalar un Cluster de Alto-Rendimiento, sin necesidad de lidiar con librerías extra requeridas ni con problemas de configuración, debido a que funciona a nivel de Kernel, añadido a la excelente interfaz openMosixView para la administración del Cluster.

El funcionamiento óptimo de openMosix se logra cuando tiene múltiples y largos trabajos ejecutando. Algunos ejemplos de este tipo de aplicaciones pueden ser instituciones financieras ejecutando análisis de riesgos o instituciones de investigación científica ejecutando comparaciones de ADN.

Sin embargo, aún hay algunas características que no incluyen las versiones actuales de openMosix, pues no todas las aplicaciones pueden migrar a otros nodos, algunas de ellas requieren funciones que se preocupen de que un proceso sea separado en un cierto contexto de sistema y usuario.

Durante los pasados años, muchas personas comenzaron a trabajar en soluciones para los problemas antes mencionados, dentro de las cuales se pueden encontrar las siguientes:

- Implementaciones que soportan Checkpointing.
- La implementación de las herramientas autodiscovery que hicieron a openMosix más fácil de instalar.
- El desarrollo del demonio General openMosix.
- El parche de límite de carga.
- El parche de la máscara del Cluster.
- La implementación de la ***Migración de Memoria Compartida***.

2.3.3.- CARACTERÍSTICAS DE UN CLUSTER OPENMOSIX

Antes de empezar a hablar de alguna de las herramientas que permiten la mejor utilización de openMosix, es necesario citar sus ventajas y desventajas.

2.3.3.1.- Ventajas de openMosix

- No se requieren paquetes extra, para el Kernel.
- No son necesarias modificaciones en el código del Kernel.

2.3.3.2.- Desventajas de openMosix

- Es dependiente del Kernel.
- No migra todos los procesos siempre, tiene limitaciones de funcionamiento.
- Problemas con memoria compartida.

Además los procesos con múltiples threads (hebras) no ganan demasiada eficiencia. Tampoco se obtendrá mucha mejora cuando se ejecute un solo proceso, como por ejemplo el

navegador [CAT2004]. Este último punto se tratará más adelante en la sección de memoria compartida y los motivos de la investigación de este tipo de Cluster, como un método de solución.

2.3.4.- TÉCNICAS DE MONITORIZACIÓN DE OPENMOSIX

Dentro de las técnicas de monitorización de openMosix existen dos, las cuales son:

- Mosmon.
- openMosixView.

2.3.4.1 Mosmon

Mosmon es la técnica de monitorización para el Cluster que ofrece la instalación básica de openMosix el cual permite ver en modo texto un gráfico de barras, con la carga que tienen los nodos pertenecientes al Cluster, ver figura Nº2.9 “Mosmon”.

Para la monitorización del Cluster, también existe otra herramienta, la cual es más amigable, esta herramienta es openMosixView.

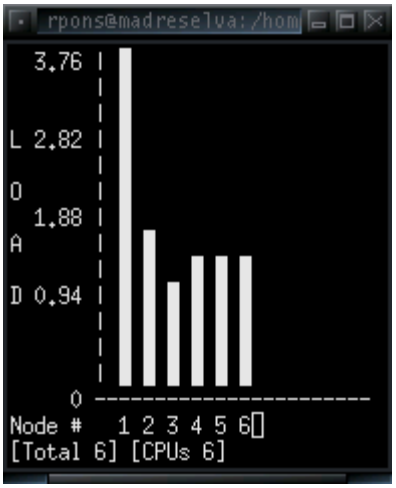

Figura Nº2.9. Mosmon.

2.3.4.2 openMosixView

La herramienta openMosixView tiene por objeto mostrar en un entorno gráfico, cuál es la carga de los nodos y además cumple el rol de interfaz para el resto de herramientas que vienen incorporadas en openMosix.

Además de presentar un entorno gráfico amigable, openMosixView permite realizar la administración del Cluster de forma fácil y dinámica, mediante las distintas herramientas que se mencionarán a continuación.

2.3.4.2.1 Herramientas de openMosixView

Las herramientas que vienen incorporadas en openMosixView son las siguientes:

- **openMosixview** es la principal aplicación la cual realiza la monitorización y administración.
- **openMosixprocs** esta aplicación es utilizada para la administración de procesos.
- **openMosixcollector** Captura la información del Cluster proporcionada por los demonios.
- **openMosixanalyzer** esta aplicación es un analizador de la información capturada por openMosixcollector.
- **openMosixhistory** historial de monitorización de procesos del Cluster.
- **openMosixmigmon** visor que representa la migración de procesos.
- **3dmosmon** visor para monitorización de datos en 3D.

Todas estas herramientas se encuentran disponibles en la pantalla principal de openMosixView.

Para la implementación de openMosixview se deben seguir las siguientes instrucciones, del manual de instalación de [CAT2004]:

Requerimientos:

- Tener instaladas las librerías QT según indique la versión,
- Derechos de superusuario -*root*-,
- Las herramientas de usuario de openMosix -*userland-tools*-.

2.3.4.2.2 Operaciones con openMosixView

La aplicación principal de openMosixView es la mostrada en la figura N°2.10.

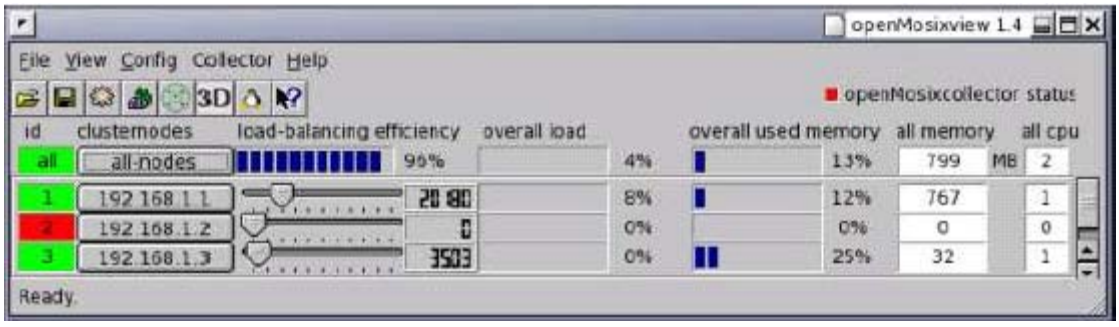

Figura N°2.10 openMosixView.

La herramienta openMosixview muestra, para cada nodo que pertenece al Cluster (cada fila): una luz, un botón, un slider, un número lcd, dos barras de progreso y un par de etiquetas.

Las luces de la izquierda muestran el ID del nodo y su estado. El color rojo significa que el nodo no se encuentra operativo y el color verde, que el nodo está operativo.

Si se hace clic en el botón que muestra la dirección IP de un nodo se llamará al diálogo de configuración, que muestra los botones para ejecutar los comandos de mosctl.

Con los sliders de velocidad se puede establecer la velocidad que considerará el Cluster para cada nodo. Este parámetro se muestra en el display lcd.

Se puede también intervenir en el balanceo de carga de cada nodo cambiando sus valores. Los procesos en un Cluster openMosix migran más fácilmente hacia un nodo cuya velocidad sea más elevada. Se tiene que tener presente que este concepto de velocidad no tiene que ser el que realmente posea la computadora, es simplemente el parámetro que se quiere que openMosix considere para cada máquina.

Las barras de progreso, que conforman el par de columnas en la mitad de la ventana, dan una idea general de la carga de cada nodo del Cluster. La primera se refiere a la carga del procesador y muestra un porcentaje que será una aproximación del valor escrito por openMosix en el fichero /proc/hpc/nodes/x/load. La segunda barra muestra la memoria utilizada en cada nodo. El box de la izquierda muestra el total de memoria disponible.

Finalmente el box del extremo derecho muestra el número de procesadores de cada nodo. En la esquina superior-izquierda se puede ver el box load-balancing efficiency, éste es un 95% y es el indicador de la eficiencia del algoritmo de balanceo de carga. Un valor próximo al 100% indica que la carga computacional ha podido dividirse equitativamente entre los nodos.

Podemos utilizar los menús de collector- y analyzer- para administrar openMosixcollector y openMosixanalyzer, respectivamente. Estas dos partes de las aplicaciones openMosixView son muy adecuadas para construir un historial del trabajo hecho y la manera en cómo se ha hecho en el Cluster.

2.3.5 MEMORIA COMPARTIDA

Como el objetivo planteado para este proyecto era el de implementar un Cluster para ser utilizado como servidor Apache, el cual trabaja con memoria compartida, nació la necesidad de investigar la posible existencia de alguna herramienta que permitiese la migración de procesos que utilizan memoria compartida y también aquellos que utilizan multi-hebras.

2.3.5.1 El Problema de la Memoria Compartida

Cuando un proceso que no utiliza memoria compartida debe ser migrado, openMosix destruye el mapa de memoria y las páginas relacionadas en el nodo hogar y hace una recreación de la misma en el nodo remoto. Esto no puede ser hecho cuando una aplicación usa memoria compartida, ya que otras aplicaciones están usando las mismas regiones de la memoria.

El resultado de la búsqueda se llama MigShm, una herramienta que debería cumplir con las funcionalidades requeridas y además poseía la característica de ser de licencia GPL.

Como se mencionó anteriormente dada las características de los Cluster lo más conveniente sería aplicar un Cluster de Alta-Disponibilidad, pero debido a que la investigación, en un principio estuvo centrada en openMosix, se consideró importante probar el funcionamiento de esta herramienta como una posible solución.

2.3.5.2 MigShm, la Solución de MAASK

Maya, Anu, Asmita, Snehal, Krushna, las desarrolladoras de MigShm, son estudiantes del Cummins College of Engineering, de la Universidad de Pune, India. Ellas estuvieron durante un año trabajando en este proyecto y escribieron un reporte acerca del mismo, en donde explican cómo lograron implementarlo.

MigShm es un parche DSM para openMosix. DSM significa Distributed Shared Memory. Este parche habilita la migración de procesos que utilizan memoria compartida en OpenMosix. Ejemplos de este tipo de aplicaciones son Apache, Xfracky, etc.

Actualmente, una de las principales limitaciones de openMosix, junto con el problema de la memoria compartida, es el de las aplicaciones que utilizan multi-hebras en el Cluster. Por consiguiente dichas aplicaciones tampoco se pueden beneficiar de las ventajas que presenta el balance de carga que es capaz de realizar openMosix. MigShm apunta a corregir ambas problemáticas, tanto la memoria compartida como las multi-hebras.

Un demonio llamado mig_shm_daemon() se ejecuta en cada nodo, el cual se encarga de manejar las migraciones de procesos que utilizan memoria compartida a los diferentes módulos del Cluster.

Apartado Nº 3 Implementaciones

3.0 IMPLEMENTACIONES.

En este apartado se explicarán las diferentes formas de llevar a cabo las implementaciones, dividiéndolas de la siguiente forma:

- Instalación de UltraMonkey en Debian Sarge.
- Instalación de openMosix.

3.1 INSTALACIÓN DE ULTRAMONKEY EN DEBIAN SARGE

Los pasos necesarios para instalar un Cluster UltraMonkey, están detallados en la Figura Nº3.l. Como se mencionó anteriormente UltraMonkey es una solución de un Cluster del tipo Alta-Disponibilidad.

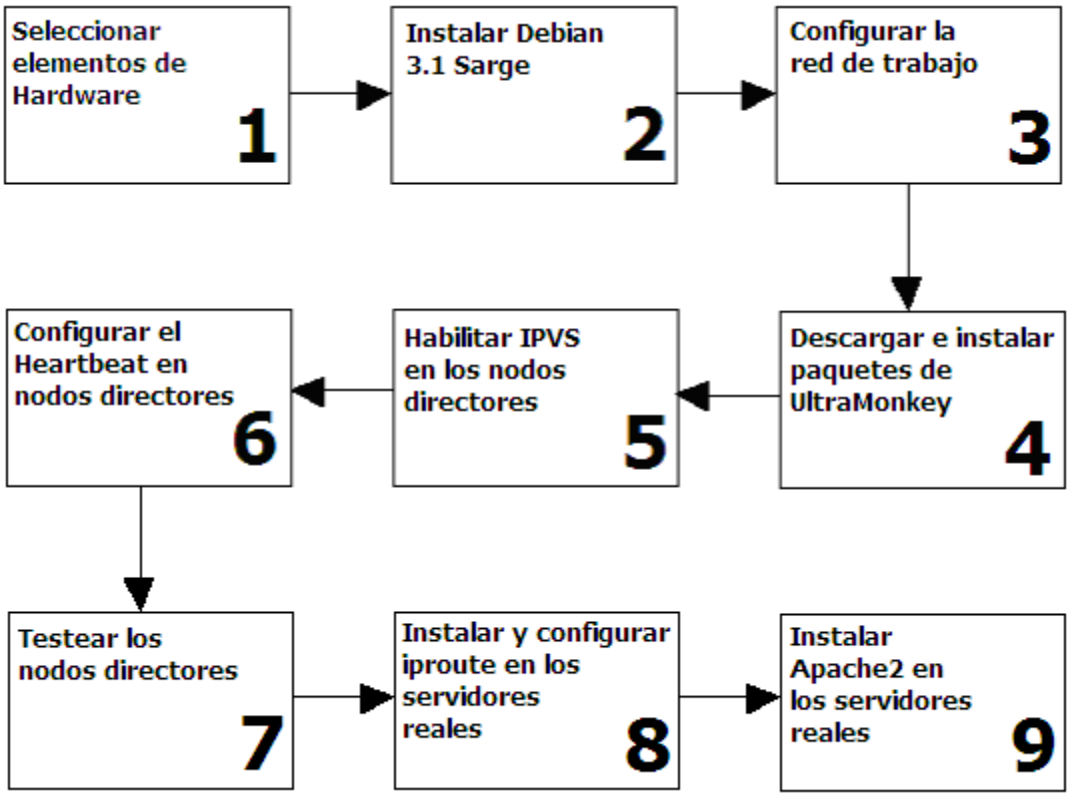

Figura Nº3.1 Esquema de Implementación de UltraMonkey.

3.1.1.- SELECCIONAR LOS ELEMENTOS DE HARDWARE

Para conseguir la instalación de UltraMonkey fue necesario utilizar un total de 4 computadores de tipo PC, localizados en la oficina Nº 11 del Departamento de Informática de la Universidad Católica del Maule (UCM). Los computadores fueron conectados por medio de un Switch Herwlett Packard HP J2601A en una red Ethernet.

Las características del nodo director están especificadas en Tabla 3.1.

debian1

Procesador	Pentium MMX, 166 Mhz
Memoria	64 MB.
Sistema Operativo	Debian Linux 3.1 (Sarge)

Tabla Nº 3.1 Características del PC debian1.

Las características de los demás nodos del Cluster son dados en la Tabla 3.2.

debian2, apache1 y apache2, son Compaq Deskpro EC series:

Procesador	Pentium III, 550Mhz
Memoria	64 MB
Sistema Operativo	Debian Linux 3.1 (Sarge)

Tabla 3.2 Características de debian2, apache1 y apache2.

Es recomendable colocar las máquinas más simples como balanceadores de carga, ya que quien realmente se ocupa de prestar los servicios son los servidores reales, y la función del balanceador es apenas traspasar los paquetes que llegan desde los clientes hacia los servidores reales.

3.1.2.- INSTALAR DEBIAN SARGE 3.1

La instalación de Debian Sarge 3.1, fue realizada con la imagen mínima de instalación que pesa 100 MB, la cual contiene el instalador y el sistema base. Para obtener otros componentes, éstos se pueden descargar desde Internet utilizando el comando apt-get.

Es posible descargar la imagen del CD de instalación de la siguiente página:

http://www.debian.org/CD/http-ftp/index.es.html

Por ahora es todo lo que se requiere para la instalación de UltraMonkey, exceptuando que es necesario instalar Apache, para montar el servidor Web.

3.1.3.- CONFIGURACIÓN RED DE TRABAJO

Como muestra la Figura Nº3.2, la red está compuesta por 4 computadores unidos mediante un switch. Las direcciones IP 192.168.0.103 y 192.168.0.104 están asignadas a los nodos directores del Cluster (debian1 y debian2) y las direcciones IP 192.168.0.101 y 192.168.0.102 a los servidores reales (apache1 y apache2). La VIP del Cluster LVS es 192.168.0.105.

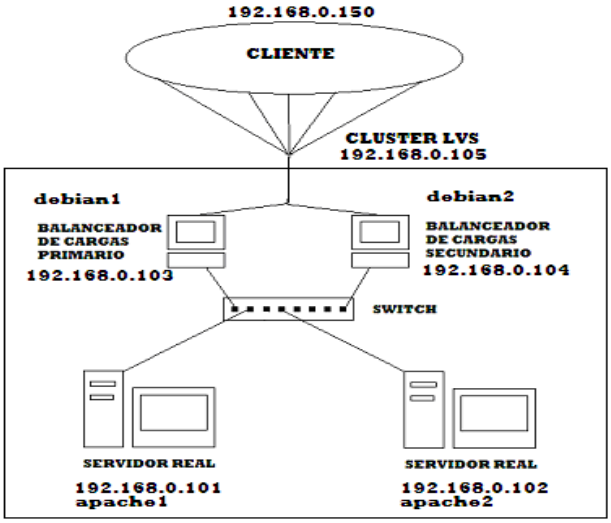

Figura Nº3.2 Esquema de red.

3.1.4.- DESCARGAR E INSTALAR PAQUETES DE ULTRAMONKEY

UltraMonkey provee paquetes adicionales y actualizados para Debian. La forma más fácil de obtener estos paquetes es usando el comando apt-get. Para hacer esto se debe añadir las siguientes líneas al comienzo del archivo /etc/apt/sources.list:

```
deb http://www.ultramonkey.org/download/3/ sarge main
```

Y luego ejecutando apt-get update para obtener la lista de los paquetes

Versiones previas de UltraMonkey requirieron de un Kernel actualizado. Sin embargo, el Kernel 2.4.27 de Sarge provee todas las capacidades que son requeridas por UltraMonkey, de tal forma que un Kernel personalizado ya no es requerido.

3.1.5 HABILITAR IPVS EN LOS DIRECTORES

Primero se debe habilitar IPVS (parche de LVS), en los directores. En debian1 y debian2 se deben ejecutar los siguientes comandos para habilitar IPVS:

```
# echo ip_vs_dh >> /etc/modules
# echo ip_vs_ftp >> /etc/modules
# echo ip_vs >> /etc/modules
# echo ip_vs_lblc >> /etc/modules
# echo ip_vs_lblcr >> /etc/modules
# echo ip_vs_lc >> /etc/modules
# echo ip_vs_nq >> /etc/modules
# echo ip_vs_rr >> /etc/modules
# echo ip_vs_sed >> /etc/modules
# echo ip_vs_sh >> /etc/modules
# echo ip_vs_wlc >> /etc/modules
# echo ip_vs_wrr >> /etc/modules
```

 # modprobe ip_vs_dh

 # modprobe ip_vs_ftp

 # modprobe ip_vs

 # modprobe ip_vs_lblc

 # modprobe ip_vs_lblcr

 # modprobe ip_vs_lc

 # modprobe ip_vs_nq

 # modprobe ip_vs_rr

 # modprobe ip_vs_sed

 # modprobe ip_vs_sh

 # modprobe ip_vs_wlc

 # modprobe ip_vs_wrr

3.1.6 CONFIGURAR EL HEARBEAT EN LOS NODOS DIRECTORES

UltraMonkey viene con paquetes adicionales. Estos paquetes sólo son requeridos para los Directores Linux que van a ejecutar Heartbeat. Pueden ser obtenidos usando el comando apt-get:

 # apt-get install ultramonkey

Otra alternativa para obtener los paquetes es desde el directorio de descargas en la página:

http://www.ultramonkey.org/download/3/debian.sarge/

Durante la instalación del paquete Ipvsadm puede que se pida configurar /etc/ipvsadm.rules, tal como se muestra en la Figura Nº3.3. Se debe seleccionar <No> ya que esto crea conflictos con la forma en que UltraMonkey configura Ipvsadm.

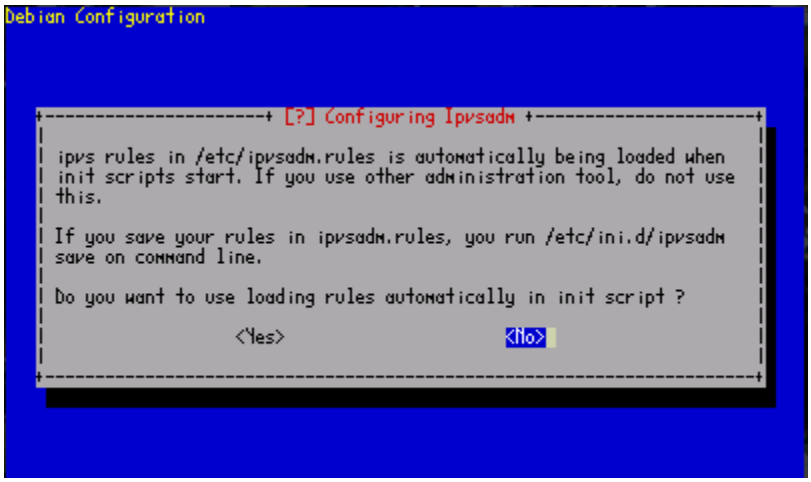

Figura N°3.3 Configuración Ipvsadm.

También puede ser necesario configurar el demonio de sincronización IPVS. Se recomienda seleccionar “none” Figura Nº3.4 “Configuración demonio de Sincronización”, ya que la configuración del demonio de sincronización no es cubierta en la documentación de UtraMonkey y en algunos casos entra en conflicto con la forma en que UltraMonkey ejecuta LVS.

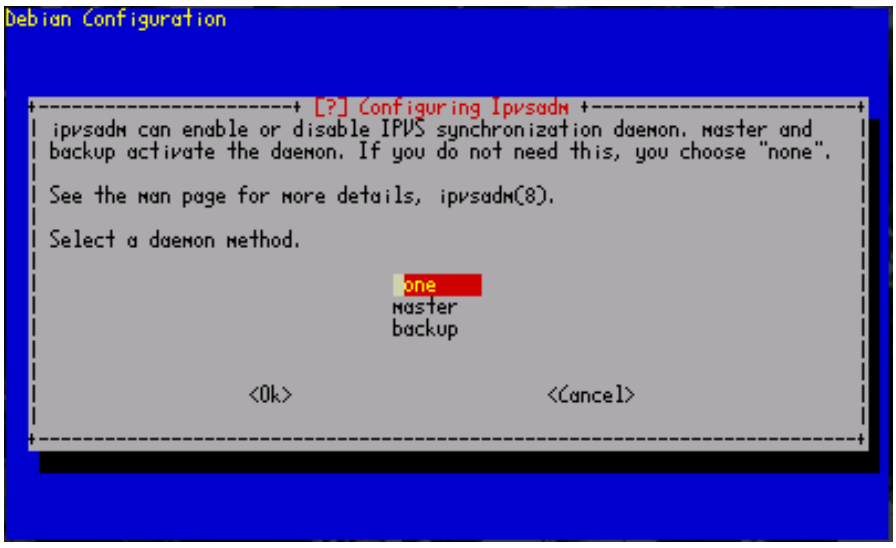

Figura N°3.4 Configuración demonio de sincronización.

3.1.6.1.-

Configuración del Cluster

Una vez que la red está configurada correctamente, se configuran los directores y los servidores reales.

3.1.6.1.1 Directores Linux

Los directores deben estar habilitados para enrutar el tráfico hacia los servidores reales. Específicamente, se debe habilitar el seguimiento IPv4. Esto se hace añadiendo a la configuración de la variable net.ipv4.i_foward a /etc/sysctl.conf como sigue:

```
net.ipv4.ip_forward = 1
```

Para que los cambios tengan efecto se debe usar el comando sysctl:

#/sbin/sysctl –p

3.1.6.1.2 Heartbeat

Para configurar el Heartbeat, deben estar instalados los archivos /etc/ha.d/ha.cf, /etc/ha.d/haresources y /etc/ha.d/authkeys.

Para visualizar el archivo /etc/ha.d/ha.cf se debe ejecutar el siguiente comando:

 # vi /etc/ha.d/ha.cf

Al ejecutar este comando el archivo que debe mostrar es el siguiente:

```
logfacility       local0
bcast       eth0              # Linux
mcast eth0 225.0.0.1 694 1 0
auto_failback off
node       debian1
node       debian2
respawn hacluster /usr/lib/heartbeat/ipfail
apiauth ipfail gid=haclient uid=hacluster
```

Para visualizar el archivo /etc/ha.d/haresources se debe ejecutar el siguiente comando:

 # vi /etc/ha.d/haresources

Al ejecutar este comando el archivo que debe mostrar es el siguiente:

```
debian1       \
ldirectord::ldirectord.cf \
LVSSyncDaemonSwap::master                                    \
IPaddr2::192.168.0.105/24/eth0/192.168.0.255
```

Los nombres de los nodos en /etc/ha.d/ha.cf y /etc/ha.d/haresources deben estar acorde a la salida que arroja el comando uname –n en cada director Linux.

Para visualizar el archivo /etc/ha.d/authkeys se debe ejecutar el siguiente comando:

 # vi /etc/ha.d/authkeys

Al ejecutar este comando el archivo que debe mostrar es el siguiente:

```
auth 33 md5 ultramonkey
```

La contraseña “ultramonkey” es utilizada en ambos demonios Heartbeat en los nodos debian1 y debian2 para autentificarse el uno al otro. Puede ser cualquier cadena de caracteres.

El /etc/ha.d/authkeys debe estar en modo 600, esto puede ser hecho usando el comando chmod.

 # chmod 600 /etc/ha.d/authkeys

El monitoreo de los servidores reales y su inserción y eliminación del conjunto de servidores seleccionables es controlado por ldirectord, el cual es ejecutado por Heartbeat. Para configurar el ldirectord, el archivo /etc/ha.d/ldirectord.cf debe estar instalado.

 # vi /etc/ha.d/ldirectord.cf

Al ejecutar este comando, el archivo que debe mostrar es el siguiente:

```
checktimeout=10
checkinterval=2
autoreload=no
logfile="local0"
quiescent=yes
virtual=192.168.0.105:80
        real=192.168.0.101:80 gate
        real=192.168.0.102:80 gate
        fallback=127.0.0.1:80 gate
        service=http
        request="ldirector.html"
```

```
receive="Pagina de Prueba"
scheduler=rr
protocol=tcp
        checktype=negotiate
```

Al recibir una petición de conexión desde un cliente, el director asigna un servidor real al cliente basado en un “schedule”. El tipo del scheduler se define en este archivo. Algunos de los schedulers disponibles son:

- rr, wrr: round robin, weighted round robin (con manejo de pesos)
- lc, wlc: least connection (menos conexiones), weighted least connection (el director tiene una tabla con el número de conexiones para cada servidor real).

Los schedulers rr, wrr, lc y wlc deberían todos funcionar de forma similar cuando el director está dirigiendo servidores reales idénticos con servicios idénticos. El scheduler lc será mejor manejando situaciones donde las máquinas son bajadas y vueltas a subir. Si los servidores reales están ofreciendo diferentes servicios y algunos tienen clientes conectados por un largo tiempo mientras otros están conectados por un corto periodo, entonces ninguno de los schedulers va a hacer un buen trabajo distribuyendo la carga entre los servidores reales. LVS no tiene ningún monitoreo de carga de los servidores reales.

Como el Cluster a implementar posee servidores reales idénticos con servicios idénticos, se optó por implementarlo con un scheduler rr (round robin).

Para asegurar que el Heartbeat se inicia y que el ldirectord no se inicia al reiniciar la máquina se usa el comando update-rc.d:

```
# /usr/sbin/update-rc.d heartbeat start 75 2 3 4 5 . stop 05 0 1 6 .
# /usr/sbin/update-rc.d -f ldirectord remove
```

Para asegurar que el ldirectord no está corriendo e inicia el Heartbeat con una nueva configuración se debe ejecutar:

 # /etc/init.d/ldirectord stop

 # /etc/init.d/heartbeat start

3.1.7 TESTEO DE LOS NODOS DIRECTORES

Luego de algunos momentos Heartbeat debería añadir la dirección IP virtual en el director Linux maestro. Para verificar que tal operación ha sido realizada se debe digitar el comando IP:

 # ip addr sh eth0

En el nodo maestro muestra:

```
4: eth0: <BROADCAST,MULTICAST,UP> mtu 1500 qdisc pfifo_fast qlen 1000
link/ether 00:50:56:4f:30:07 brd ff:ff:ff:ff:ff:ff
inet 192.168.0.103/24 brd 192.168.6.255 scope global eth0
inet 192.168.0.105/24 brd 192.168.0.255 scope global secondary eth0
```

En el nodo en espera muestra:

```
2: eth0: <BROADCAST,MULTICAST,UP> mtu 1500 qdisc pfifo_fast qlen 1000
link/ether 00:16:3e:50:e3:3a brd ff:ff:ff:ff:ff:ff
inet 192.168.0.104/24 brd 192.168.0.255 scope global eth0
```

Heartbeat debe también ejecutar ldirectord para configurar LVS en este nodo. Para comprobar que ldirectord está corriendo se debe usar:

 # /usr/sbin/ldirectord ldirectord.cf status

En el nodo maestro muestra:

```
ldirectord for /etc/ha.d/ldirectord.cf is running with pid: 1455
```

En el nodo en espera muestra:

```
ldirectord is stopped for /etc/ha.d/ldirectord.cf
```

Para inspeccionar la tabla actual del Kernel LVS, el comando Ipvsadm debe ser utilizado. Un ejemplo de la invocación a dicho comando que muestra que ldirectord encuentra todos los servidores disponibles se presenta a continuación:

#/sbin/ipvsadm -L -n

El archivo debería mostrar lo siguiente:

```
IP Virtual Server version 1.2.1 (size=4096)
Prot LocalAddress:Port Scheduler Flags
  -> RemoteAddress:Port           Forward Weight ActiveConn InActConn
TCP  192.168.0.105:80 rr
  -> 192.168.0.101:80             Route   0      0          0
  -> 192.168.0.102:80             Route   0      0          0
  -> 127.0.0.1:80                 Local   1      0          0
```

Al añadir direcciones IP virtuales y ejecutar ldirectord, el Heartbeat debería también ejecutar el demonio de sincronización de LVS. El demonio maestro debería estar ejecutándose en el actual director Linux maestro, y el demonio de respaldo en el otro director Linux. Se puede verificar esto inspeccionando el estado del recurso LVSSyncDaemonSwap.

En el director Linux maestro

 # /etc/ha.d/resource.d/LVSSyncDaemonSwap master status

```
master running
(ipvs_syncmaster pid: 1689)
```

En los directores Linux en espera

 # /etc/ha.d/resource.d/LVSSyncDaemonSwap master status

```
master stopped
(ipvs_syncbackup pid: 1563)
```

3.1.8.- INSTALAR Y CONFIGURAR IPROUTE EN LOS SERVIDORES REALES

Finalmente debemos configurar los nodos apache1 y apache2 para que acepten peticiones en la IP virtual 192.168.0.105

En los nodos apache1 y apache2 ejecutamos:

 # apt-get install iproute

 # vi /etc/sysctl.conf

El archivo que debería mostrar es el siguiente:

```
net.ipv4.conf.all.arp_ignore = 1
net.ipv4.conf.eth0.arp_ignore = 1
net.ipv4.conf.all.arp_announce = 2
net.ipv4.conf.eth0.arp_announce = 2
```

Luego se debe ejecutar los siguientes comandos:

 # sysctl –p

 # vi /etc/network/interfaces

El archivo a mostrar, es el mostrado en el siguiente recuadro:

```
auto lo
iface lo inet loopback
auto lo:0
iface lo:0 inet static
 address 192.168.0.105
 netmask 255.255.255.255
 pre-up sysctl -p > /dev/null
```

Luego se debe ejecutar el comando:

 # Ifup lo:0

Para verificar que la interfaz se levantó, se usa el comando ifconfig. La siguiente salida ha sido truncada sólo para mostrar información para lo:0.

```
ip addr sh lo
1: lo:  mtu 16436 qdisc noqueue
 link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
          inet 127.0.0.1/8 scope host lo
          inet 192.168.0.105/32 scope global lo:0
```

Notar que la máscara de red 255.255.255.255 en la interfaz lo:0 indica que ésta sólo va a aceptar tráfico 192.168.0.105.

Ahora creamos el archivo ldirector.html. Este archivo es requerido por los dos balanceadores de carga repetidamente para ver si los dos nodos apache aún están corriendo.

Asumiendo que el directorio raíz de los documentos de apache es /var/www, se crea el archivo /var/www/ldirector.html ejecutando el comando:

 # vi /var/www/ldirector.html

El archivo puede contener cualquier cosa para efectos de esta investigación el utilizado es el siguiente :

```
Pagina de Prueba
```

Para que los cambios tengan efecto el trabajo de red debe ser reiniciado:

/etc/init.d/networking restart

```
Reconfiguring network interfaces: done.
```

3.1.9.- INSTALAR APACHE 2 EN LOS SERVIDORES REALES

Finalmente es necesario instalar Apache 2 en los servidores reales para esto es necesario ejecutar el siguiente comando:

 # apt-get install apache2

De esta forma se han detallado todos los pasos necesarios para la implementación de un Cluster de Alta-Disponibilidad y balanceo de carga UltraMonkey.

3.2 INSTALACIÓN DE OPENMOSIX

Esta instalación, como se mencionó en apartados anteriores, no es la solución que en esta investigación se propone, pero es considerada e incluida debido a que en ella se invirtió una gran cantidad de tiempo.

El siguiente esquema (Figura Nº3.5) se muestran los pasos necesarios para instalar un Cluster openMosix.

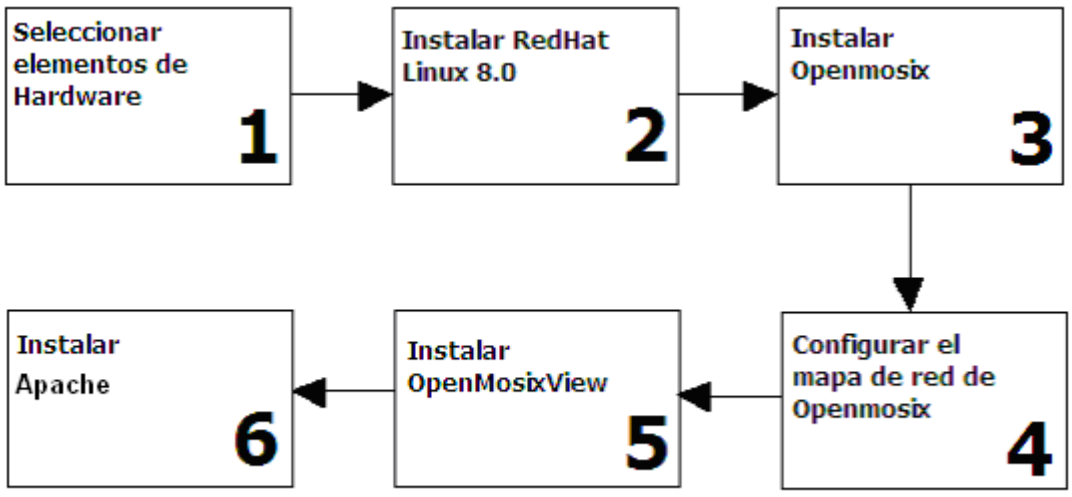

Figura Nº3.5 Esquema de Instalación de openMosix

3.2.1 SELECCIONAR ELEMENTOS DE HARDWARE

Para esta instalación fue necesaria la utilización de dos PC cuyas características están dadas en la Tabla 3.3 y la Tabla 3.4.

Además se contó con un Hub ethernet 10/100 de 8 bocas para conectar los nodos entre si.

Las características de Mosix1 son :

Procesador	Pentium IV, 1,6 Ghz.
Memoria	256 MB.
Disco Duro	40 GB.

Tabla 3.3 Características del PC Mosix1.

Las características de Mosix2 son:

Procesador	Pentium IV, 1,8 Ghz.
Memoria	256 MB.
Disco Duro	40 GB.

Tabla 3.4 Características del PC Mosix2.

Los PC mencionados son equipos mucho más potentes que los utilizados en la implementación de UltraMonkey, no por que sean necesarios equipos más potentes sino porque esta implementación se llevó a cabo en los equipos personales de los investigadores de esta memoria.

3.2.2 INSTALAR RED HAT LINUX 8.0

El sistema operativo instalado en los equipos fue Red Hat Linux 8.0. El cual ofrece la posibilidad de realizar la instalación de forma rápida utilizando paquetes binarios RPM.

La instalación de Red Hat Linux 8.0 es muy sencilla ya que presenta una interfaz bastante amigable, lo que hace más fácil su instalación.

La siguiente lista contiene los paquetes básicos a instalar en nuestro sistema, y que se incluyen en los CD's de instalación de Red Hat Linux 8.0:

- gcc
- kernel-package
- libc6-dev
- libncurses5-dev
- fakeroot
- bin86

Para efectos de esta investigación, se optó por trabajar bajo el entorno gráfico GNOME, y con el gestor de arranque LILO. También se puede utilizar el gestor de arranque GRUB, ya que la instalación a través de paquetes RPM hace la modificación automática una vez que se instala el Kernel de openMosix.

3.2.3 INSTALAR OPENMOSIX

Para la instalación de openMosix, tenemos dos opciones: puede ser a través de paquetes binarios o a través de archivos fuentes.

Los paquetes binarios, los archivos fuentes de instalación de openMosix y las herramientas de usuario están disponibles para su descarga gratuita en la siguiente dirección:

http://sourceforge.net/project/showfiles.php?group_id=46729.

A continuación se detallan las dos formas de instalar openMosix, una a través de paquetes binarios y otra utilizando los archivos fuentes, siendo esta última más engorrosa debido a que se debe compilar el Kernel. En la instalación utilizando los archivos fuentes se detalla la instalación del parche Migshm el cual sirve para que los procesos con memoria compartida puedan ser balanceados dentro del Cluster.

3.2.3.1 Instalación de openMosix utilizando paquetes binarios

Este tipo de instalación es bastante sencilla, sólo es necesario ejecutar las siguientes instrucciones, luego de haber descargado obviamente los paquetes correspondientes:

 # rpm -i openMosix-kernel-2.4.20-openMosix3.i686.rpm

Luego es necesario realizar la configuración del gestor de arranque. En el caso de GRUB, el paquete binario hace la configuración automáticamente. Para LILO, se debe modificar el archivo agregando la siguiente entrada:

```
Image=/boot/kernelopenMosix
        label=openMosix
        root=/dev/hda1
        initrd=/boot/initrd.img **
        read-only
```

Luego se reinicia la máquina con la nueva instalación de openMosix y se ejecuta el resto de los paquetes RPM con las instrucciones:

 # rpm -i openMosix-tools-0.2.4-1.i686.rpm
 # rpm -i openMosixview-1.5-redhat72.i686.rpm

Una vez ejecutadas estas instrucciones, se cuenta con todo lo necesario para empezar a configurar el Cluster openMosix.

3.2.3.2 Instalación de openMosix utilizando archivos fuentes

A través de esta instalación es posible instalar el parche Migshm, el cual permite trabajar con procesos que utilizan memoria compartida.

El primer paso para esta instalación, es descomprimir las fuentes del Kernel de Linux, luego de haberlas descargado, para esto ejecutamos:

 # tar xzvf linux-2.4.26.tar

Luego movemos los parches al directorio del Kernel de Linux y lo parchamos con el comando:

 # patch -Np1 < 'nombre_parche_de_openmosix'
 # patch -p1 < 'nombre_parche_de_migshm'

Una vez modificado, el Kernel debe ser alojado en el directorio /usr/src, para ello ejecutamos:

 # cd ..

 # mv linux-2.4.26 /usr/src/

Existen 3 formas de configurar el Kernel:

 # make oldconfig
 # make menuconfig
 # make xconfig

Cualquiera fuere el método escogido, el Kernel debe ser configurado activando las siguientes opciones:

OpenMosix
[*] openMosix process migration support
[ ] Support clusters with a complex network topology
[*] Stricter security on openMosix ports
(1) Level of process-identity disclosure (0-3)
[*] Poll/Select exceptions on pipes
[ ] Disable OOM Killer
[*] Shared memory migration support (Experimental)
[*] flush() support for consistency
[*] Kernel Debug messages
[ ] Enable Extension: Local Time

Adicionalmente, se debe habilitar la tarjeta de red según corresponda. Una vez hecho esto, se debe guardar la configuración de Kernel. Para ello se ejecutan los siguientes comandos:

 # make dep
 # make clean
 # make bzImage
 # make modules
 # make modules_install
 # make install

Durante la experimentación, se obtuvo el siguiente error de instalación:

Subject: grubby fatal error: unable to find a suitable template

Este error no está relacionado a alguna falla en la configuración ni en la compilación del Kernel, sino que indica que, o bien, la creación de la imagen initrd del Kernel ha fallado, o el gestor de arranque no fue configurado automáticamente.

En caso de presentarse este error, se deben seguir las instrucciones que se detallan a continuación.

3.2.3.2.1 Construir una imagen initrd

Una imagen initrd es necesaria para cargar los módulos SCSI al momento del arranque del sistema.

Para construir una nueva imagen initrd, debemos ejecutar /sbin/mkinitrd con los parámetros de la forma en que se muestra a continuación:

 # /sbin/mkinitrd /boot/initrd-version_del_kernel.img version_del_kernel

En el ejemplo, /boot/initrd-version_del_kernel.img es el nombre del archivo de la nueva imagen initrd, y version_del_kernel es la version del Kernel cuyos módulos (se puede ver en /lib/modules) deben ser usados en la creación de la misma.

Así, para efectos de esta investigación, ejecutamos el comando:

 # /sbin/mkinitrd /boot/initrd-2.4.26-migshm.img 2.4.26-migshm

Una vez compilado e instalado nuestro Kernel, se debe configurar el gestor de arranque según corresponda y reiniciar el sistema.

Debemos repetir esta instalación en cada máquina que funcionará como nodo de nuestro Cluster.

3.2.3.2.2.- Instalación de las Herramientas de Usuario

Los pasos necesarios para la instalación de las herramientas de usuario de openMosix, desde las fuentes, son los que se mencionan a continuación.

Primeramente, se debe descomprimir el archivo de fuente:

 # tar xvzf openmosix-tools-0.3.6-2.tar

Luego hay que entrar en el directorio creado y ejecutar las siguientes instrucciones:

 # ./configure
 # make
 # make install

Una vez finalizada la instalación, es posible remover los archivos binarios y objetos dejados por la instalación con el comando:

 # make clean

Hecho esto, se ha finalizado la instalación de las herramientas necesarias para monitorear y controlar el Cluster openMosix + MigShm.

3.2.4 CONFIGURACIÓN DEL MAPA DE RED OPENMOSIX

Como muestra la Figura Nº3.6, la red está compuesta por 2 computadores unidos mediante un hub. Las direcciones IP 192.168.10.1 y 192.168.10.2 están asignadas a los nodos componentes del Cluster (Mosix1 y Mosix2).

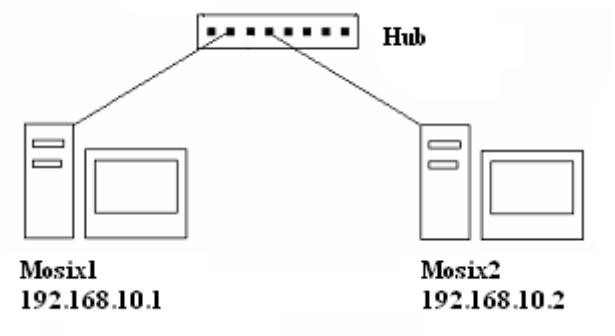

Figura Nº3.6 Configuración Red openMosix.

Una vez instaladas todas las herramientas, se debe configurar el mapa de openMosix. Para esto, existen 2 métodos:

- Editar manualmente el archivo de configuración /etc/openMosix.map
- Usar el demonio de autodiscovery Omdiscd (herramienta de usuario en modo texto).

A continuación se detallan los dos métodos existentes para configurar el mapa de openMosix.

3.2.4.1- Configuración mediante omdiscd

El proceso omdiscd está incluido en el paquete openMosix-tools. Al ejecutar el comando omdiscd en uno de los computadores de la red, éste envía un paquete broadcast a la red, en el cual se da aviso de que se ha incorporado un nuevo nodo al Cluster.

3.2.4.2- Configuración manual

Para realizar una configuración manual del mapa de openMosix, se debe modificar el archivo openMosix.map del directorio /etc. Esta implementación estuvo compuesta por dos máquinas, las cuales tienen por IP las direcciones 192.168.10.1 y 192.168.10.2. Así, el archivo openMosix.map debería lucir de la siguiente forma:

```
# openMosix CONFIGURATION
# ==================
# openMosix-# IP  number-of-nodes
# ============================
1  192.168.10.1  1
2  192.168.10.2  1
```

El primer número corresponde al número de nodo del Cluster, seguido por la dirección IP del nodo, y el número total de nodos dentro del rango de dicha IP.

La mayoría de las instalaciones con Red Hat tienen un detalle extra que arreglar ya que a menudo se obtiene el siguiente error:

```
[root@mosix1 root]# /etc/init.d/openMosix start
Initializing OpenMosix...
setpe: the supplied table is well-formatted,
but my IP address (127.0.0.1) is not there!
```

Esto significa que el nombre de host no está definido en el archivo /etc/hosts con la misma IP que en el archivo openMosix.map.

Una vez terminada la instalación y configuración de openMosix, se deben reiniciar todos los nodos del Cluster, y de esta forma se estará a un paso de contar con un Cluster funcionando.

Luego de reiniciar los nodos, se puede comenzar con la monitorización del Cluster, para lo cual se pueden utilizar las herramientas de usuario en modo texto, o bien, utilizar openMosixView que trabaja en modo gráfico.

3.2.5 INSTALAR OPENMOSIXVIEW

La herramienta openMosixView está diseñada para monitorear el Cluster openMosix a través de una interfaz gráfica que la hace “amigable” para el usuario. Por lo mismo, se recomienda su utilización. Dicha herramienta es llamada a través del siguiente comando:

```
# openMosixview
```

Los paquetes RPM tienen como directorio de instalación la ruta /usr/local/openMosixview-<versión>.

Luego de descargar la última versión de los paquetes RPM de openMosixView, es necesario instalarlos como cualquier otro paquete, desde la línea de comandos:

 # rpm -i openMosixview-<versión>-<distribución>.rpm

Esto instalará los ficheros binarios en el directorio /usr/bin/. Para desinstalarlos se debe ejecutar:

 # rpm -e openMosixview

El paquete binario de instalación de openMosixView, el cual contiene las herramientas de usuario para modo gráfico, debe ser descargado desde el siguiente enlace:

http://www.openMosixview.com/download.html.

El paquete descargado en este proyecto es:

openMosixview-1.5-redhat72.i386.rpm

3.2.6 INSTALACIÓN DE APACHE

El paquete binario de Apache utilizado es el que incluye el disco de instalación de Red Hat Linux 8.0, por lo cual sólo basta con incluirlo en la instalación de Linux, o bien, instalar el RPM posteriormente.

3.3.- COSTOS DE EQUIPOS

Para la implementación, ya sea en openMosix o en UltraMonkey, se utilizaron computadores ya dados de baja de los laboratorios de la Universidad Católica del Maule.

Como se mencionó anteriormente los computadores utilizados son, en su gran mayoría Compaq Deskpro, con un procesador Pentium III de 550 Mhz. Éstos en el mercado actual en su calidad de equipos de segunda mano, los podemos encontrar desde $60.000, que es sumamente bajo, hasta los $110.000.

Para comprobar esta información, se puede visitar la siguiente página:

http://www.deremate.cl

Con respecto al switch, los valores de éstos están entre los $16.899 hasta los $49.990. Por lo tanto sigue siendo bastante económica esta implementación en cuanto al costo de los equipos.

En comparación con el Cluster, adquirir un servidor con dos o más procesadores usados con características similares a los PC en que se implementaron los Cluster está entre los $189.000, llegando a sumas superiores al millón de pesos, en el caso de servidores nuevos y con tecnologías actuales.

Otra característica y ventaja importante de un Cluster UltraMonkey con respecto un servidor con dos o más procesadores está en que con el Cluster podemos garantizar Alta-Disponibilidad en caso de, por ejemplo, ocurrir un corte de luz. Aun en ese escenario es posible seguir trabajando con otra fuente generadora de corriente, a un director y un servidor real, por lo tanto las ventajas de UltraMonkey son variadas y altamente aplicables.

Apartado Nº 4

Pruebas

4.0 PRUEBAS

En este apartado se especificarán las pruebas realizadas a las diferentes implementaciones, y ver si éstas responden bien a las exigencias requeridas. Este apartado está divido de la siguiente manera:

- Pruebas a UltraMonkey:
  1) Realizar varias peticiones de servicio a la dirección IP virtual.
  2) Desconectar el director primario.
  3) Desconectar el servidor real.

- Pruebas a openMosix.
  1) Testeo de balance de carga del algoritmo de la serie de Fibonacci.
  2) Testeo de balance de carga de Apache2.

A continuación se detalla cada una de ellas.

4.1 PRUEBAS CON ULTRAMONKEY

Para probar el correcto funcionamiento de esta implementación se realizaron las siguientes 3 pruebas:

- Prueba Nº1 a UltraMonkey: Realizar varias peticiones de servicio a la dirección IP virtual.
- Prueba Nº2 a UltraMonkey: Desconectar el director primario debian1
- Prueba Nº3 a UltraMonkey: Desconectar el servidor apache1

Para conocer el estado del Cluster en cada prueba, se utilizó la herramienta Ipvsadm.

En condiciones normales, con ambos servidores apache funcionando correctamente, la salida de Ipvsadm es la siguiente:

 # /sbin/ipvsadm -L -n

```
IP Virtual Server version 1.2.1 (size=4096)
Prot LocalAddress:Port Scheduler Flags
  -> RemoteAddress:Port           Forward Weight ActiveConn InActConn
TCP  192.168.0.105:80 rr
  -> 192.168.0.101:80             Route   1      0          0
  -> 192.168.0.102:80             Route   1      0          0
```

La primera línea muestra la versión de la herramienta Ipvsadm es la versión de la misma, en este caso 1.2.1. En la cuarta línea se muestra el protocolo utilizado (TCP), la dirección IP virtual del Cluster (192.168.0.105) y el algoritmo de escalonamiento utilizado (rr = round-robin). Luego se muestran las direcciones IP de los servidores reales, seguido del peso (Weight), número de conexiones activas (ActiveConn) y número de conexiones inactivas de cada uno (InActConn).

4.1.1 PRUEBA Nº1 A ULTRAMONKEY

Realizar varias peticiones de servicio a la dirección IP virtual.

Como muestra la Figura Nº4.1, la red está compuesta por 4 computadores unidos mediante un switch. Las direcciones IP 192.168.0.103 y 192.168.0.104 están asignadas a los nodos directores del Cluster (debian1 y debian2) y las direcciones IP 192.168.0.101 y 192.168.0.102 a los servidores reales (apache1 y apache2). La VIP del Cluster LVS es 192.168.0.105.

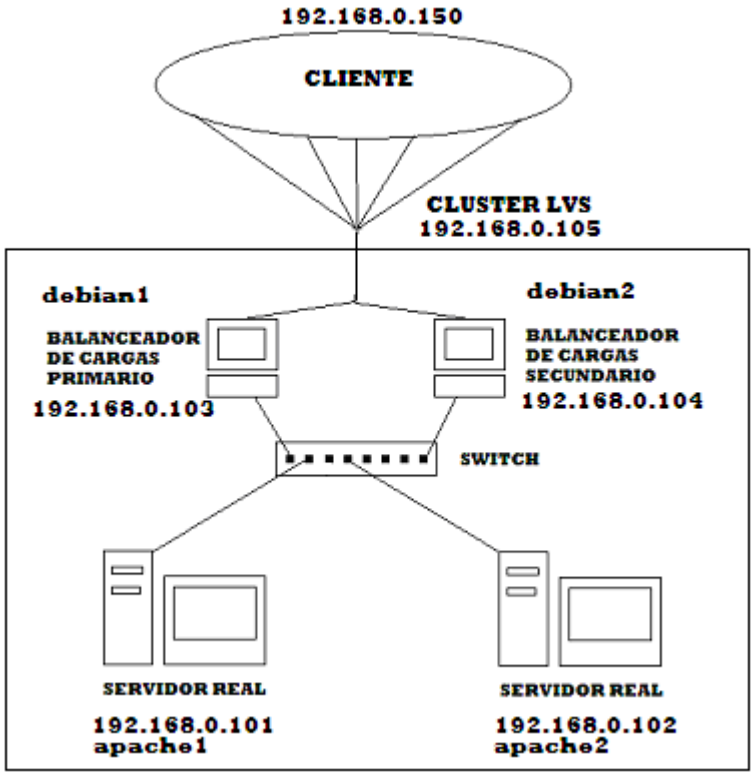

Figura Nº4.1 Esquema del Cluster.

Para probar si las peticiones eran traspasadas por los directores a los servidores reales, y la forma en que la carga se balanceaba, se realizaron seis peticiones de servicio desde un computador cliente. Al usar la herramienta Ipvsadm después de realizar estas peticiones el resultado mostrado fue el siguiente:

 # /sbin/ipvsadm -L –n

```
IP Virtual Server version 1.2.1 (size=4096)
Prot LocalAddress:Port Scheduler Flags
  -> RemoteAddress:Port           Forward Weight ActiveConn InActConn
TCP  192.168.0.105:80 rr
  -> 192.168.0.101:80             Route   1      0          3
  -> 192.168.0.102:80             Route   1      0          3
```

Se observa que el Ipvsadm muestra 3 conexiones inactivas para cada servidor real. Eso se debe a que el tiempo de consulta es insignificante y, por lo mismo nunca muestran las conexiones como activas. Si se tratase de un TELNET, se mostrarían las conexiones activas hasta que se finalice la sesión. También se observa que la carga se distribuye en forma homogénea entre los servidores reales al tratarse de un escalonamiento por round-robin en el cual ambos tienen el mismo peso (weitht).

El resultado obtenido fue el esperado, por lo cual el resultado de esta prueba fue exitoso.

4.1.2 PRUEBA Nº2 A ULTRAMONKEY

Desconectar el director primario debian1.

Como muestra la Figura Nº4.2, la red está compuesta por 3 computadores unidos mediante un switch. La IP 192.168.0.104 está asignada al nodo director del Cluster debian2 y las direcciones IP 192.168.0.101 y 192.168.0.102 a los servidores reales (apache1 y apache2). La VIP del Cluster LVS es 192.168.0.105. En este caso se desconecta el director debian1.

Para comprobar la Alta-Disponibilidad del Cluster, es decir, el correcto funcionamiento del Heartbeat operando en los nodos directores debian1 y debian2, se desconectó de la red el director primario debian1. Luego se realizó una nueva petición del servicio virtual desde el computador cliente para comprobar la disponibilidad del mismo.

Antes de desconectar debian1, se ejecutó el siguiente comando en el director en espera debian2:

 # /etc/ha.d/resource.d/LVSSyncDaemonSwap master status

```
master stopped
```

Esto indica que el director debian2 está detenido.

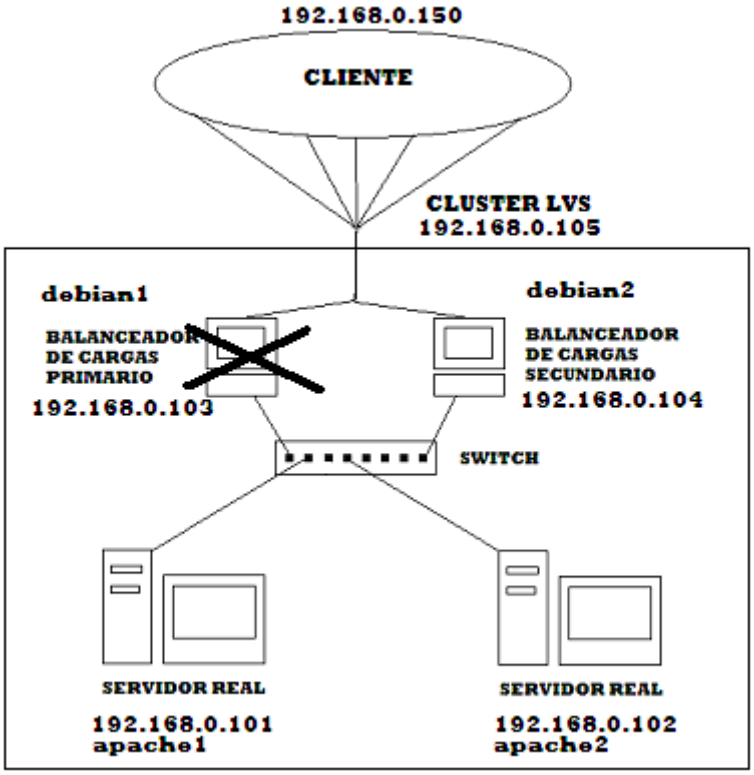

Figura Nº4.2 Cluster sin director primario.

Luego se desconectó el director debian1 y se volvió a ejecutar el comando en debian2 y arrojó lo siguiente:

 # /etc/ha.d/resource.d/LVSSyncDaemonSwap master status

```
master running
```

Esto indica que el director en espera debian2 entró en funcionamiento cuando fue desconectado debian1. Para comprobar su correcto funcionamiento se hizo una nueva petición a la dirección virtual desde el cliente y el resultado fue correcto.

El resultado obtenido en la segunda prueba fue el esperado y se concluye que ésta fue exitosa.

4.1.3 PRUEBA Nº 3 A ULTRAMONKEY

Desconectar el servidor apache1.

Como muestra la Figura Nº4.3, la red está compuesta por 3 computadores unidos mediante un switch. Las direcciones IP 192.168.0.103 y 192.168.0.104 están asignadas a los nodos directores del Cluster (debian1 y debian2) y la dirección IP 192.168.0.102 al servidor real apache2. La VIP del Cluster LVS es 192.168.0.105. En este caso se desconecta el servidor real apache1.

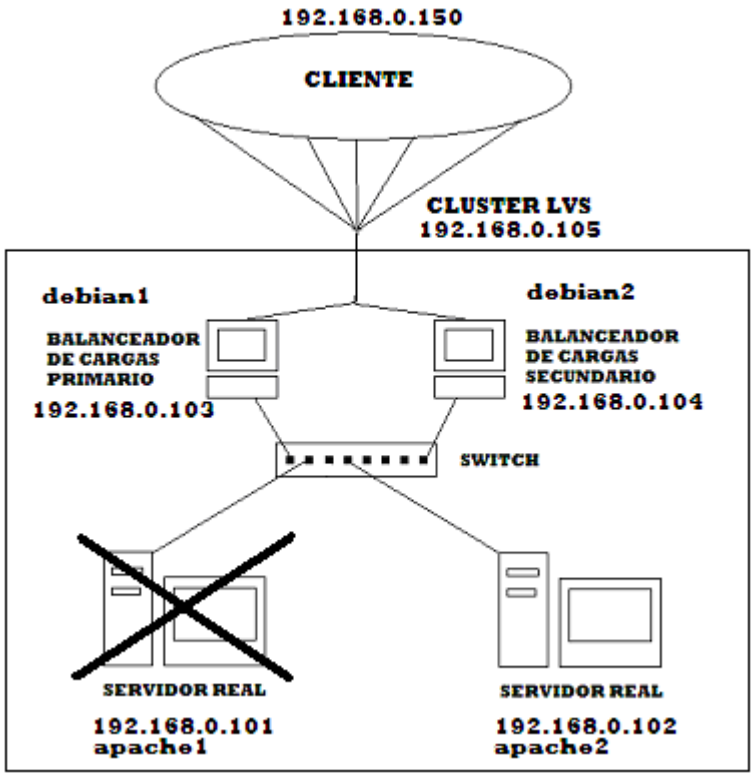

Figura Nº4.3 Cluster con servidor real apache1 desconectado.

Para comprobar el monitoreo que los directores deben hacer de los servidores reales, a fin de saber en todo momento aquellos que están operativos, se desconectó el servidor apache1.

A fin de saber desde el computador cliente cuál servidor real era aquel que respondía efectivamente la petición del servicio virtual, se creó una pagina index.html en apache1 y apache2. En apache1 la pagina contenía el string “servidor apache 1” y en apache2 el string “servidor apache 2”.

Las consultas al servicio virtual se hicieron directamente pidiendo el archivo index.html.

Antes de desconectar el servidor apache1, se realizaron varias consultas al archivo index.html del servicio virtual. Como era de esperarse, en la ventana del navegador del computador cliente se fue mostrando alternadamente en cada consulta el mensaje “servidor apache 1” y “servidor apache 2”, con lo cual comprobamos una vez más que las consultas son traspasadas correctamente por los directores hacia ambos servidores reales.

Luego desconectamos apache1 y ejecutamos una vez más el comando Ipvsadm:

 # /sbin/ipvsadm -L -n

```
IP Virtual Server version 1.2.1 (size=4096)
Prot LocalAddress:Port Scheduler Flags
  -> RemoteAddress:Port           Forward Weight ActiveConn InActConn
TCP  192.168.0.105:80 rr
  -> 192.168.0.102:80             Route   1      0          3
```

Se observa que sólo aparece como nodo el servidor apache2 (192.168.0.102), lo cual es el resultado esperado. Luego se ejecutaron consultas desde el cliente a la pagina index.html del servicio virtual y se pudo visualizar en la ventana del explorador el mensaje “servidor apache 2”. Con esto se comprobó que el servicio era atendido íntegramente por el servidor activo apache2.

El resultado obtenido en la tercera prueba fue el esperado y se concluye que la prueba fue exitosa.

4.2.- PRUEBAS A OPENMOSIX

Una vez instalado el Cluster openMosix, el equipo de investigación programó una serie de pruebas para el mismo. Estas pruebas consistieron en ejecutar un mismo proceso varias veces en segundo plano, de tal forma de comprobar si la carga era o no balanceada.

El proceso escogido fue el famoso algoritmo de la serie de Fibonacci, el cual fue programado para ejecutarse hasta un cierto límite de ciclos, de tal forma de comparar los tiempos de ejecución tanto cuando el Cluster se encontraba operativo, como cuando se ejecutaba con un nodo aislado.

Los resultados de esta prueba fueron alentadores, ya que al ejecutar el programa de Fibonacci cuatro veces en el nodo aislado el tiempo requerido fue justamente el doble al requerido cuando el nodo 2 se agregaba al Cluster. La misma prueba se repitió varias veces, encontrando siempre un resultado idénticamente exitoso.

El primer resultado que se obtuvo de la utilización del Cluster openMosix fue que efectivamente se lograba el balanceo de carga mediante la migración de procesos completos a los demás nodos. Sin embargo, al ejecutar sólo una vez una aplicación, ésta se ejecutaba íntegramente en el nodo raíz, sin migrar, lo cual dio algunas luces acerca del problema que se presentaría posteriormente en la investigación.

El objetivo principal de esta investigación, fue desde un principio implementar un Cluster que tuviese algún tipo de utilidad importante, más específicamente, un Cluster que fuese capaz de realizar el balanceo de carga de un servidor Web Apache. En un comienzo se pensó que con implementar el Cluster openMosix sería suficiente, pero no fue así.

4.2.1- PRUEBAS CON APACHE Y OPENMOSIX

Para probar el servidor apache instalado, se recurrió al apache benchmark tool. Esta herramienta es capaz de simular una cierta cantidad de consultas bajo un cierto nivel de

concurrencia. El archivo ejecuta operaciones de cálculo intensivo, además de operaciones de entrada y salida a archivos.

La forma de utilización de esta herramienta consiste en ejecutar la siguiente instrucción:

 # ab -c 50 -n 500 http://192.168.1.28/test.php

De la anterior instrucción se entiende que se ejecutan 500 consultas bajo un nivel de concurrencia de 50, al servidor cuya dirección IP es 192.168.1.28

Los resultados obtenidos ejecutando esta prueba bajo openMosix no fueron exitosos, ya que, como se ha explicado, no realiza la migración de procesos que utilizan memoria compartida. La principal falla se encuentra en procesamiento paralelo.

Al realizar la prueba bajo openMosix junto a MigShm, la prueba falló ya que el nodo raíz colapsó al ejecutar la instrucción.

En todo el tiempo de investigación dedicado a openMosix (incluido el parche Migshm) no conseguimos que los procesos que utilizan memoria compartida migraran a otros equipos, por lo que esta solución fue descartada.

4.3.- TABLAS DE SELECCIÓN

En esta sección se detalla mediante tablas el proceso utilizado, para llevar a cabo la selección de la solución más adecuada a los objetivos planteados.

En la Tabla 4.1 nos muestra los distintos Clusters existentes y cuales fueron los seleccionados para la investigación.

Solución	Seleccionada
Kimberlite	No
openMosix	Sí
MPI	No
PVM	No
UltraMonkey	Sí
Beowulf	No

Tabla 4.1 Selección de Soluciones a implementar.

En la tabla 4.2 se indica la cantidad de pasos necesarios para llevar a cabo la instalación de la solución seleccionada.

Solución	Nº de Pasos
openMosix	6
UltraMonkey	9

Tabla 4.2 Número de pasos de la solución.

La Tabla 4.3 muestra la cantidad de computadores necesarios para realizar la implementación.

Solución	Nº de Pc
openMosix	2
UltraMonkey	4

Tabla 4.3 Computadores necesarios para implementar la solución.

En la Tabla 4.4 se indica si las soluciones realizan balanceo de carga.

Solución	Balancea Carga
openMosix	Sí
UltraMonkey	Sí

Tabla 4.4 Balanceo de Carga.

En la Tabla 4.5 se indica si las soluciones realizan balanceo de carga del servidor Apache.

Solución	Balancea Carga Apache
openMosix	No
UltraMonkey	Sí

Tabla 4.5 Balanceo de Carga de Apache.

CONCLUSIONES

Los resultados obtenidos en las pruebas realizadas en esta investigación corroboran que es posible implementar un Servidor Web Apache en un Cluster de Alta-Disponibilidad utilizando Linux y construido a base de equipos de bajo costo, lo cual, si bien no la hace la implementación mas conveniente en cuanto a rendimiento, si permite que sea una alternativa viable en términos del costo de los equipos para pequeñas y medianas empresas que no cuentan con los recursos para adquirir una gran computadora.

Como se mostró, a pesar de necesitar menos pasos y computadores, la solución otorgada por openMosix no cumple con los requisitos necesarios para llevar a cabo este proyecto al no poder balancear la carga del servidor Apache. Cabe mencionar que ni siquiera con la inclusión del parche Migshm, el cual maneja la memoria compartida, se convierte en una solución viable.

En contraste, la solución de UltraMonkey utilizando LVS, mostró que es capaz de conseguir el correcto funcionamiento de un servicio de Alta-Disponibilidad y el balanceo de carga. Se puede decir además que por tratarse de un Cluster, sus costos en comparación con los de una supercomputadora son considerablemente más bajos.

Por todo lo mencionado anteriormente se puede afirmar que UltraMonkey brinda una solución viable al problema planteado, considerando tanto los aspectos económicos, como funcionales al haber aprobado satisfactoriamente todas las pruebas realizadas, cumpliendo así con el objetivo de esta investigación.

Así, se ha logrado la implementación de un Cluster de Alta-Disponibilidad y Balanceo de Carga.

Tanto el código abierto como los Clusters van ganando cada vez más terreno en la actualidad dentro del mundo de las tecnologías de información y no cabe duda que en los próximos años su implantación en las diferentes organizaciones será mucho más común de lo que es hoy y, por ende, muchos más se verán beneficiados por sus grandes atributos.

BIBLIOGRAFÍA

[BRO2004] Brown, Robert G
Duke University, Physics Department
Book: Engineering a Beiowulf-style Computer Cluster.
http://www.phy.duke.edu/rgb

[CAT2004] CATALÁN, MIQUEL
http://www.urjc.es/cat/hidra/manuales/openMosix/howTo-openMosixES_0.4beta.pdf

[CAT2004a] CATALÁN, MIQUEL
http://es.tldp.org/Manuales-LuCAS/doc-manual-openMosix-1.0/doc-manual-openMosix-1.0.pdf

[CHI2003] CHIRINOV, ROUMEN
Proyecto Cluster openMosix (Linux)
http://www.noticias3d.com/articulo.asp?idarticulo=248&pag=2

[GAR2003] GARCÍA LEIVA, RAFAEL A.
Universidad Autónoma de Madrid.
http://es.tldp.org/Manuales-LuCAS/doc-instalacion-cluster-alta-disponibilidad/instalacion-cluster-alta-disponibilidad/

[HER2005] HERNÁNDEZ VÁSQUEZ, MIGUEL
http://www.elai.upm.es/spain/Investiga/GCII/personal/mhernandez/pagina_personal_de_Miguel_Hernandez.htm

[LVS2005] Página oficial del proyecto Linux Virtual Server (LVS)
http://www.linuxvirtualserver.org/

[OLE2004] OLEA, ISMAEL.
Introducción a los Cluster de computadoras
http://es.tldp.org/Manuales-LuCAS/doc-cluster-computadoras/doc-cluster-computadoras-html/node7.html

[RID2003] RIDRUEJO, FRANCISCO; AGIRRE, JON Y ALONSO, JOSÉ MIGUEL
http://www.sc.ehu.es/acwmialj/papers/jornadas03_a.pdf

[WIK2004] WikiLearning, Manual de openMosix 2004
http://www.wikilearning.com/clusters_ha_ii-wkccp-9756-15.htm

[WIK2004b] Wikipedia, Balanceo de Carga
http://es.wikipedia.org/wiki/Balanceo_de_carga

[WIK2005] Wikipedia, Cluster de la Alta disponibilidad 2005
http://es.wikipedia.org/wiki/Cluster_de_alta_disponibilidad

[WIK2006] Wikipedia, Linux Virtual Server 2006.
http://es.wikipedia.org/wiki/Linux_Virtual_Server

[WIK2006b] Wikipedia, Escalabilidad 2006
http://es.wikipedia.org/wiki/Escalabilidad

[WIK2006c] Wikipedia, Alto rendimiento
http://es.wikipedia.org/wiki/Cluster_de_alto_rendimiento

GLOSARIO

ARP

Son las siglas en inglés de Address Resolution Protocol (Protocolo de resolución de direcciones). Es un protocolo de nivel de red responsable de encontrar la dirección hardware (Ethernet MAC) que corresponde a una determinada dirección IP.

Beowulf

Es una clase de computador masivamente paralelo de altas prestaciones principalmente construido a base de un Cluster de componentes hardware estándar. Se interconecta mediante una red privada de gran velocidad. Generalmente se compone de un grupo de PCs o estaciones de trabajo dedicados a ejecutar tareas que precisan una alta capacidad de cálculo.

DNS

Domain Name System (DNS) es una base de datos distribuida y jerárquica que almacena información asociada a nombres de dominio en redes como Internet. Aunque como base de datos el DNS es capaz de asociar distintos tipos de información a cada nombre, los usos más comunes son la asignación de nombres de dominio a direcciones IP y la localización de los servidores de correo electrónico de cada dominio.

DSM

Distributed Shared Memory (DSM, o memoria distribuida compartida) es un tipo de implementación hardware y software, en la que cada nodo de un Cluster tiene acceso a una amplia memoria compartida que se añade a la memoria limitada privada, no compartida, propia de cada nodo.

fwmarks

Es un método flexible que mediante marcas de firewall (fwmarks) permite administrar de manera más simple y flexible un elevado número de servicios. Se puede utilizar para agrupar

servicios, de forma que estos fueran tratados conjuntamente por un mismo servidor real. De esta manera, LVS puede utilizar el método de agrupamiento de puertos.

GNU/Linux

Es la denominación defendida por Richard Stallman y otros para el sistema operativo que utiliza el Kernel Linux en conjunto con las aplicaciones de sistema creadas por el proyecto GNU. Comúnmente este sistema operativo es denominado como Linux, aunque ésta denominación no es correcta.

GPL

(General Public License o licencia pública general) es una licencia creada por la FreeSoftware Foundation a mediados de los 80, y está orientada principalmente a los términos de distribución, modificación y uso de software. Su propósito es declarar que el software cubierto por esta licencia es software libre.

HTTP

El protocolo de transferencia de hipertexto (*HTTP*, HyperText Transfer Protocol) es el protocolo usado en cada transacción de la Web (WWW).

IA32

Es la arquitectura de microprocesadores de 32 bits de Intel (*Intel Architecture* 32). Son los microprocesadores más usados en los ordenadores personales (PC).

IPv4

Protocolo utilizado para designar una dirección IP mediante un número binario de 32 bits.

Kernel

Es la parte fundamental de un sistema operativo. Es el software responsable de facilitar a los distintos programas acceso seguro al hardware de la computadora o en forma más básica, es el encargado de gestionar recursos, a través de servicios de llamada al sistema.

Loopback

El dispositivo de red **loopback** es un interfaz de red virtual que siempre representa al propio dispositivo independientemente de la dirección IP que se le haya asignado. El valor en IPv4 es 127.0.0.1. y ::1 para el caso de IPv6.

MAC

En redes de computadoras la dirección MAC (Media Access Control address) es un identificador hexadecimal de 48 bits que se corresponde de forma única con una tarjeta o interfaz de red. Es individual, cada dispositivo tiene su propia dirección MAC determinada y configurada por el IEEE (los primeros 24 bits) y el fabricante (los últimos 24 bits).

Mosctl

Es la principal utilidad para la configuración de openMosix.

MPI

La Interfaz de Paso de Mensajes (conocido ampliamente como MPI, siglas en inglés de Message Passing Interface) es un protocolo de comunicación entre computadoras. Es el estándar para la comunicación entre los nodos que ejecutan un programa en un sistema de memoria distribuida.

NAT

Network Address Translation (Traducción de Dirección de Red) es un mecanismo utilizado por routers IP para intercambiar paquetes entre dos redes que se asignan mutuamente direcciones incompatibles. Consiste en convertir en tiempo real las direcciones utilizadas en

los paquetes transportados. También es necesario editar los paquetes para permitir la operación de protocolos que incluyen información de direcciones dentro de la conversación del protocolo.

Proxy

Es un programa o dispositivo que realiza una acción en representación de otro. La finalidad más habitual es la del servidor proxy, que sirve para permitir el acceso a Internet a todos los equipos de una organización cuando sólo se puede disponer de un único equipo conectado, esto es, una única dirección IP.

PVM

La Máquina Virtual Paralela (conocida como PVM por sus siglas en inglés de Parallel Virtual Machine) es una librería para el cómputo paralelo en un sistema distribuido de computadoras. Está diseñado para permitir que una red de computadoras heterogénea comparta sus recursos de cómputo (como el procesador y la memoria ram) con el fin de aprovechar esto para disminuir el tiempo de ejecución de un programa al distribuir la carga de trabajo en varias computadoras.

Red Hat Linux

Es una distribución Linux creada por Red Hat, que fue una de las más populares en los entornos de usuarios domésticos.

SCSI

Es el acrónimo en inglés de Small Computer System Interface, es un interfaz estándar para la transferencia de datos entre distintos dispositivos del bus de la computadora.

Sheduler

Es un componente funcional muy importante de los sistemas operativos multitarea y multiproceso, y es esencial en los sistemas operativos de tiempo real. Su función consiste en

repartir el tiempo disponible de un microprocesador entre todos los procesos que están disponibles para su ejecución.

SMP

Es el acrónimo de Symmetric Multi-Processing, multiproceso simétrico. Se trata de un tipo de arquitectura de ordenadores en que dos o más procesadores comparten una única memoria central.

Sockets

Designa un concepto abstracto por el cual dos programas (posiblemente situados en computadoras distintas) pueden intercambiarse cualquier flujo de datos, generalmente de manera fiable y ordenada. Un socket queda definido por una dirección IP, un protocolo y un número de puerto.

SSI

Es acrónimo del inglés Small-Scale Integration (integración a baja escala) y hace referencia a los primeros circuitos integrados que se desarrollaron. Cumplían funciones muy básicas, como puertas lógicas y abarcan desde unos pocos transistores hasta una centena de ellos.

Tunneling

Es una técnica que consiste en encapsular un protocolo de red sobre otro (protocolo de red encapsulador) creando un túnel dentro de una red de comunicaciones (o red de computadoras). El uso de esta técnica persigue diferentes objetivos, dependiendo del problema que se esté tratando, como por ejemplo la comunicación de islas en escenarios multicast, la redirección de tráfico.

VIP

Dirección IP virtual, por donde los computadores clientes realizan la petición de un servicio a un LVS.

WAN

Es una red de área amplia, acrónimo de la expresión en idioma inglés ***W**ide **A**rea **N**etwork*, es un tipo de red de computadoras capaz de cubrir distancias desde unos 100 hasta unos 1000 km, dando el servicio a un país o un continente.