Universidad de las Ciencias Informáticas
Facultad 7

Cluster de servidores web para aplicaciones desarrolladas sobre software libre que soportan altos niveles de concurrencia

Trabajo de diploma para optar por el título de
Ingeniero en ciencias informáticas

Autor: Adrián Misael Peña Montero

Tutor: Ing. Juan Carlos Pujol García

Asesor: Ing. Juenlis Enrique Coss Piña

Ciudad de La Habana, Julio de 2008
"Año 50 de la Revolución"

DECLARACIÓN DE AUTORÍA

Declaro que soy el único autor de este trabajo y autorizo a la Facultad 7 de la Universidad de las Ciencias Informáticas y a la Empresa Softel a hacer uso del mismo en su beneficio.

Para que así conste firmo la presente a los 9 días del mes de Julio del año 2008.

Autor: _____

Adrián Misael Peña Montero

Tutor: _____

Juan Carlos Pujol García

DATOS DE CONTACTO

Juan Carlos Pujol García: Ingeniero en Máquinas Computadoras, ISPJAE-1986, Especialista Superior en Informática de Softel. Profesor Auxiliar de la UCI (Universidad de las Ciencias Informáticas), Maestrante de telemática. Labora actualmente como jefe del proyecto de Servicios Remotos de la empresa Softel, su grupo que se dedica a la administración remota de servidores y aplicaciones informáticas, tanto basadas en plataforma libre como propietaria. Imparte la asignatura Sistemas de Bases de Datos en la UCI. Está interesado en: Técnicas que incrementen la seguridad (en el sentido amplio) de los servidores y aplicaciones informáticas, técnicas de monitoreo de funcionamiento y de eficiencia, técnicas de gestión de ancho de banda, VPN, teleinformática, administración (para alta eficiencia) de servidores de bases de datos.

Empresa: Softel
Dirección: Softel, Carretera a San Antonio Km. 2 1/2 Reparto Torrens, Infraestructura productiva de la UCI, Ciudad Habana.
Teléfono: (53-7) 835-8258
E-mail: juanca@softel.cu, juanca@uci.cu

Juenlis Enrique Coss Piña: Ingeniero Informático, Universidad de Holguín, 2006, Especialista de la Dirección de Teleformación. Profesor de Teleinformática. Facultad 10, Universidad de Ciencias Informáticas. Es miembro de un grupo de investigación y desarrollo orientado al desarrollo y soporte de herramientas para la teleformación en la Universidad de Ciencias Informáticas. Miembro de la comunidad Moodle en Cuba. Ha participado en un gran número de eventos nacionales e internacionales. Ha publicado trabajos en eventos y revistas a nivel nacional e internacional. Ha cumplido en varias ocasiones misión en el extranjero.

Universidad de las Ciencias Informáticas
Dirección: Carretera a San Antonio Km. 2 1/2 Reparto Torrens, Dirección de Teleformación, Ciudad Habana.
Teléfono: (53-7) 837-2453
E-mail: juenlis@uci.cu

Dedicado a mi familia por haberme dado su amor y apoyo en todo momento.

RESUMEN

El proceso de informatización que se está llevando a cabo en Cuba, ha creado la necesidad de desarrollar un grupo de aplicaciones médicas para automatizar varias de las actividades cotidianas del sector de Salud Pública. La mayoría de las aplicaciones desarrolladas son de tipo web, su implementación se realiza utilizando *software* libre y su explotación se lleva a cabo de forma centralizada.

El objetivo de la investigación es diseñar una propuesta de solución que permita obtener tiempos de respuestas aceptables en los servidores web sobre los que se ejecutan las aplicaciones médicas. Esta se basa en la utilización de un cluster de balanceo de carga.

La hipótesis de la investigación es que las aplicaciones médicas cubanas, de tipo web, que soportan altos niveles de concurrencia pueden ser ejecutadas sobre un cluster de balanceo de carga para servidores web, obteniendo tiempos de respuesta aceptables sin que se afecte su funcionamiento. La hipótesis fue contrastada y como resultado se obtuvo el diseño genérico y algunas variantes específicas de una propuesta de solución que permite aumentar la capacidad de procesamiento en los servidores web.

La propuesta de solución se diseñó para utilizar el balanceador de carga IPVS combinado con el *software* Ldirectord y Heartbeat. Algunos de las variantes específicas utilizan además el *software* NFS.

Palabras claves

Cluster, balanceo de carga, servidores web, pruebas, concurrencia, tiempo de respuesta, capacidad de procesamiento, alto rendimiento, *software* libre.

ÍNDICE

Pág.

INTRODUCCIÓN

El proceso de informatización que se está llevando a cabo en Cuba ha desencadenado el desarrollo de un gran número de aplicaciones para automatizar las actividades cotidianas que se realizan en los diferentes sectores y esferas de la sociedad. El Ministerio de Salud Pública (MINSAP) es uno de los sectores priorizados en este proceso. Para él ya se han desarrollado un grupo de aplicaciones médicas y se trabaja en el desarrollo de nuevas versiones y aplicaciones. En el desarrollo de estas aplicaciones participa principalmente la Facultad 7 de la Universidad de las Ciencias Informáticas y la empresa productora de *software* Softel.

Siguiendo las políticas que se han dictado en el país sobre el uso del *software* libre y la utilización de este en el desarrollo de las aplicaciones para el proceso de informatización, la mayoría de las aplicaciones médicas que se están desarrollando o ya están en explotación utilizan software libre y se despliegan sobre servidores con sistema operativo Linux en alguna de sus distribuciones.

La generalidad en el diseño de las aplicaciones médicas, es que sean de tipo web para facilitar su explotación de forma centralizada. Esto se debe a la necesidad de almacenar y procesar, en conjunto, la información proveniente de las instituciones o entidades que están distribuidas geográficamente por todo el país.

Una consecuencia de la explotación centralizada de las aplicaciones médicas de tipo web, es el acceso a ellas de un gran número de usuarios de manera concurrente. También existen otras aplicaciones médicas de tipo web, que debido a las características de la entidad u organización que las requiere, tendrán que soportar altos niveles de concurrencia, aún sin ser explotadas de forma centralizada.

Las aplicaciones que son utilizadas por muchos usuarios de manera concurrente, generan grandes volúmenes de carga para los servidores web y de bases de datos sobre los que se ejecutan, provocando que se requiera *hardware* con altas capacidades de procesamiento.

Debido al *hardware* existente para el despliegue de las aplicaciones médicas, la capacidad de procesamiento de los servidores web sobre los que se ejecutan varias de estas es inferior a la requerida y aunque el *hardware* puede ser mejorado, existe un límite.

Brindar los servicios utilizando servidores web con capacidad de procesamiento inferior al que requieren las aplicaciones, repercute directamente en el rendimiento del sistema y puede afectar su correcto funcionamiento. En estas condiciones, el sistema, puede alcanzar tiempos de respuestas inaceptables para las peticiones que realizan los usuarios o colapsar.

Esta investigación se realiza para dar solución al problema que constituye obtener tiempos de respuestas aceptables en los servidores web sobre los que se ejecutan las aplicaciones médicas cubanas y tiene como objeto de estudio los servidores web antes mencionados.

Dado el problema a resolver, el objetivo de la investigación es diseñar una propuesta de solución que permita obtener tiempos de respuestas aceptables en los servidores web sobre los que se ejecutan las aplicaciones médicas y el campo de acción es el tiempo de respuesta de dichos servidores.

Para lograr el objetivo de la investigación se planteó como hipótesis que: las aplicaciones médicas cubanas que soportan altos niveles de concurrencia pueden ser ejecutadas sobre un cluster de balanceo de carga para servidores web, obteniendo tiempos de respuesta aceptables y sin que se afecte su funcionamiento.

Durante el desarrollo de la investigación se identificaron las aplicaciones médicas desarrolladas sobre *software* libre por la Facultad 7 de la UCI y por Softel que pueden necesitar este tipo de soluciones y se crearon grupos con las semejantes a partir de sus características. Después se escogió una aplicación representativa de cada uno de los grupos y se diseñaron variantes de la propuesta de solución para ellas. Para garantizar que las variantes de la propuesta de solución cumplen su objetivo se le realizaron un grupo de pruebas a cada una.

Para dar cumplimiento al objetivo antes planteado se definieron un grupo tareas que fueron efectuadas durante la investigación, estas tareas son las siguientes:

1. Realizar un estudio de las técnicas de cluster de balanceo de carga existentes actualmente y el *software* que se utiliza para su implementación.
2. Identificar los tipos de pruebas necesarios para la validación de las propuestas de soluciones basadas en cluster de balanceo de carga.

3. Identificar un grupo de aplicaciones médicas de tipo web, desarrolladas sobre software libre por la Facultad 7 de la UCI y Softel, que soporten altos niveles de concurrencia y puedan necesitar de este tipo de soluciones.
4. Seleccionar una muestra representativa del grupo de aplicaciones que permitan estudiar el comportamiento de la generalidad, al desplegarse sobre un cluster de balanceo de carga para servidores web.
5. Diseñar variantes de propuesta de solución, basadas en cluster de balanceo de carga, para cada una de las aplicaciones de muestra.
6. Incluir alta disponibilidad, confiabilidad y facilidades de escalabilidad, siempre que sea posible, en cada una de las variantes de propuesta de solución.
7. Realizar las pruebas necesarias a cada una de las variantes de propuesta de solución que permitan su validación.
8. Diseñar un plan de seguridad básico y monitoreo para las variantes de propuesta de solución.

Como resultado de este trabajo se espera obtener el diseño de un conjunto de variantes de la propuesta de solución, que permitan obtener tiempos de respuesta aceptables en los servidores web sobre los que se ejecutan las aplicaciones médicas cubanas de los grupos identificados.

Este documento se estructuró en cuatro capítulos. El primero de estos se centra en conceptos relacionados con los cluster de balanceo de carga y las descripción del *software* que utiliza para su implementación. En el segundo capítulo está orientado a identificar las aplicaciones médicas de tipo web que pueden necesitar este tipo de solución y a la selección de la muestra que se utilizará en la investigación. El capítulo tres se dedico al diseño de la propuesta de solución y el cuarto a la descripción y análisis de resultados de las pruebas realizadas para validarla.

CAPÍTULO 1. FUNDAMENTACIÓN TEÓRICA

En este capítulo se estudian las técnicas de cluster de balanceo de carga existentes actualmente, el *software* que se utiliza para su implementación y las pruebas que se pueden realizar para validar una propuesta de solución basada en este tipo de cluster.

En él también se describen otros tipos de software y herramientas que será utilizadas durante la realización de las pruebas o para la gestión del monitoreo y la seguridad de la propuesta de solución.

1.1. Técnicas para disminuir los tiempos de respuesta de los servidores web

Para obtener tiempos de respuestas aceptables de los servidores web sobre los que se ejecutan las aplicaciones que soportan un elevado número de usuarios concurrentes, es necesario aumentar su capacidad de procesamiento. Para lograrlo existen varias soluciones o técnicas. A continuación se muestran algunas utilizadas para servidores web.

Utilizar arquitecturas que permitan despliegue en varios niveles

Las aplicaciones web que se implementan con una arquitectura que permite la separación en capas y el despliegue en varios niveles, utilizando un servidor para cada nivel, logran aumentar la capacidad de procesamiento del sistema. Esta arquitectura se utiliza por otras razones, pero al separar una aplicación web en pequeños subsistemas que colaboran entre sí, que incluso pueden encontrarse separados geográficamente, se realiza una distribución del procesamiento. Como resultado agregado se logra que disminuya el número de tareas en cada servidor y aumente la capacidad de procesamiento del sistema completo (Pujol García et al. 2007).

Esta técnica tiene el inconveniente de solo poder ser utilizada en aplicaciones web que han sido desarrolladas con una arquitectura de este tipo.

Mejorar el *hardware* de los servidores

Mejorar el *hardware* de los servidores aumentando la capacidad de procesamiento, la memoria RAM (*Random Access Memory*) y la velocidad de acceso a disco y de las interfases de red es una solución

muy común pero que tiene un límite práctico, que una vez alcanzado es muy difícil de superar. El límite puede estar dado por el desarrollo tecnológico del momento, la escalabilidad o la relación entre el costo y utilidad. Frecuentemente es uno de los dos últimos el que primero se alcanza (Pujol García et al. 2007).

Utilizar técnicas de *clustering*

Un cluster de servidores es un sistema paralelo o distribuido, utilizado como un único recurso computacional, que consiste en un conjunto de ordenadores conectados entre sí a través de una red de alta velocidad (Kopper 2005). Generalmente se crea usando computadoras comunes de bajo costo y se utiliza para tareas que una sola no tiene capacidad de realizar o las realiza de una manera poco eficiente.

Dependiendo de las características y el objetivo para el cual fue concebido, un cluster puede clasificarse de las siguientes formas:

- Alto rendimiento

  Un cluster de alto rendimiento (*High Performance*) se utiliza para conseguir altas prestaciones en cuanto a capacidad de cálculo. El funcionamiento básico de este tipo de cluster consiste en reducir el tamaño de una tarea en un conjunto de tareas más pequeñas y repartirla entre todos sus nodos. Así logra resolver una tarea en el menor tiempo posible consiguiendo un alto rendimiento en el procesamiento de datos (Mark et al. 2000).

  Para desplegar una aplicación sobre este tipo de cluster, la misma debe hacer sido desarrollada teniendo en cuanta los requerimientos que estos imponen.

- Alta disponibilidad

  Un cluster de alta disponibilidad (*High Availability*) se basa en la redundancia de recursos y se utiliza para servicios que necesitan estar disponibles las 24 horas del, día los siete días de la semana. Este tipo de cluster se compone por al menos dos servidores. En su forma más básica un servidor, el primario, se encuentra prestando el servicio y los otros, los secundarios, se mantienen supervisándolo para si falla, alguno tome su lugar sin afectar el servicio (Sloan 2004).

- Balanceo de carga

  Un cluster de balanceo de carga (*Load Balancing*) se utiliza en entornos donde es necesario realizar muchas tareas pequeñas. Se compone por dos tipos de nodos, los balanceadores de carga y los servidores reales. Su principio de funcionamiento es repartir las tareas entre todos los servidores reales que componen el cluster (Sloan 2004). Este tipo de cluster es altamente escalable y su arquitectura facilita realizar implementaciones de alta disponibilidad.

Para aumentar la capacidad de procesamiento en un servidor web, una de las técnicas más utilizadas, es sustituirlo por un cluster de balanceo de carga. Sin embargo, la solución final generalmente es una combinación parcial o total de algunas de estas técnicas.

1.2. Cluster de balanceo de carga

Con el crecimiento explosivo de Internet el tráfico en la red de redes aumentó rápidamente, provocando un crecimiento significativo en el volumen de trabajo de los servidores. Como consecuencia los servidores que soportaban sitios muy populares se sobrecargaron (Bourke 2001). Para superar el problema de la sobrecarga en los servidores se utilizaron principalmente dos soluciones.

La primera solución consistía en mejorar el *hardware* de los servidores en busca de mayor rendimiento. Esta constituía una solución temporal, ante el continuo aumento en el volumen de trabajo y conducía a un proceso cíclico y complejo de mejoras al *hardware* (Linux Virtual Server Project 2005c). Otros inconvenientes de esta solución son la poca escalabilidad y el costo.

El principio de la segunda solución era construir un cluster de servidores. De esta forma, cuando aumentaba el volumen de trabajo, se comenzaba a añadir servidores al cluster hasta disminuir los niveles de carga. Así, se obtenía una solución más rentable y mucho más escalable (Linux Virtual Server Project 2005c).

Antes que el balanceo de carga fuera una tecnología o un producto viable, los administradores de redes utilizaban un proceso de distribución de carga conocido como DNS (*Domain Name System*) *round robin* (Bourke 2001). Este se basaba en una facilidad de los servidores de DNS, que permite asociar más de una dirección IP (*Internet Protocol*) a un mismo nombre de dominio.

Implementar un balanceo de carga basado en DNS *round robin* es muy sencillo, solo se necesita configurar el servidor de DNS para que cada vez que se le pida resolver un nombre de dominio especifico, responda utilizando una dirección IP distinta (Bourke 2001). Las direcciones IP que utiliza el servidor de DNS para resolver el nombre de dominio, son las de los servidores que prestan el servicio que se está balanceando.

El principal inconveniente de este método de balanceo de carga, es que el servidor de DNS no conoce el estado de los servidores que prestan el servicio. Como consecuencia, si un servidor falla y el DNS responde a las peticiones de algunos clientes con su dirección IP, no podrán acceder al servicio.

Los balanceadores de carga se utilizan para distribuir la carga entre los nodos de un cluster de servidores, logrando que los servicios que brindan cada uno de ellos, puedan aparecer como un único servicio brindado por una sola dirección IP. De esta manera se logra que los clientes interactúen con el cluster como si se tratara de un solo servidor (Linux Virtual Server Project 2005c).

Un balanceador de carga realiza el balanceo a nivel de conexiones. Así se obtiene un mayor equilibrio en la distribución de carga entre los servidores, que el obtenido utilizando DNS *round robin*. Además, la utilización de balanceadores de carga permite tener servidores fuera de servicio sin afectar el funcionamiento del sistema (Linux Virtual Server Project 2005c).

Funcionamiento de un cluster de balanceo de carga.

El proceso normal de acceder a un servicio de red comienza con la realización, por el cliente, de una petición a un servidor y termina con la respuesta del servidor ofreciendo el servicio solicitado (Martínez Jiménez et al. 2003). Para que el cliente pueda realizar una petición al servidor necesita establecer una conexión con el, mediante esta se realiza el intercambio de información, por lo que es necesario mantenerla hasta que el servidor termine de enviar la respuesta.

Cuando se sustituye un servidor por un cluster de balanceo de carga, la solicitud de conexión que realiza el cliente llega directamente al balanceador de carga. El balanceador de carga decide a que servidor real reenviarla y a partir de este momento todos los paquetes pertenecientes a esa conexión son reenviados al mismo servidor real. Si el cliente necesita establecer otra conexión, el balanceador de carga puede reenviarla al mismo o a otro servidor real.

Un cluster de balanceo de carga se comporta como un servidor virtual y es completamente transparente a sus clientes. Para lograrlo, presta servicios por una sola dirección IP (Martínez Jiménez et al. 2003). Esta dirección IP pertenece al balanceador de carga y constituye el punto de acceso al cluster, normalmente se le conoce como dirección IP virtual (VIP, por sus siglas en inglés) (Bourke 2001).

El balanceador de carga es un dispositivo de red que se sitúa entre los clientes y los servidores reales. Así logra centralizar la recepción de solicitudes de conexión, para que no lleguen directamente a los servidores reales. Además de la VIP tiene al menos otra dirección IP para comunicarse con el resto de los nodos del cluster.

Entre las funciones del balanceador de carga se encuentra: decidir el servidor real con el que se va ha establecer la conexión, proceso que realiza utilizando algoritmos de distribución carga y mantener la comunicación entre los clientes y los servidores reales (Martínez Jiménez et al. 2003). En la figura 1.1 se muestra la ubicación del balanceador de carga y la VIP en un cluster.

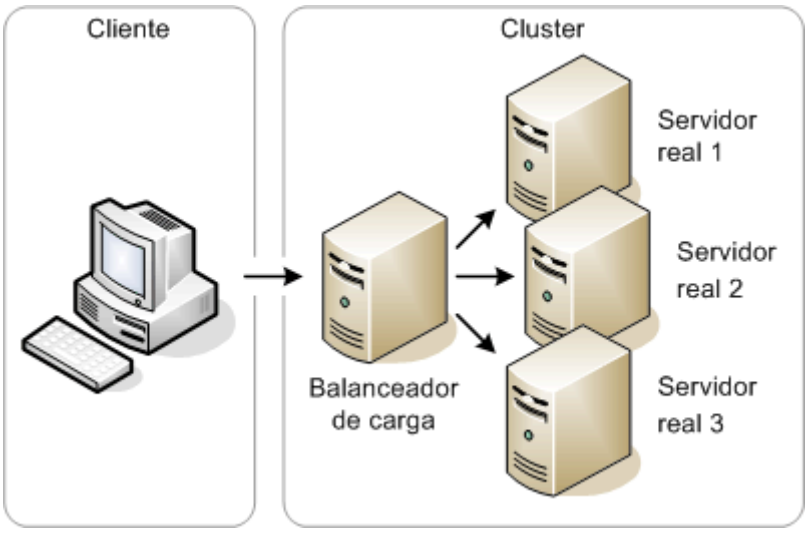

Figura 1.1. Ubicación del balanceador de carga en el cluster

Los servidores reales son los nodos de procesamiento del cluster. En ellos es donde se prestan los servicios a los que acceden los clientes. Normalmente utilizan una dirección IP y un puerto asociado para prestar los servicios (Bourke 2001).

Algoritmos de distribución de carga

Los balanceadores de carga disponen de un grupo de algoritmos para la distribución de las conexiones entre los servidores reales. Algunos pueden ser más efectivos que otros en dependencia del servicio que se esté balanceando (Martínez Jiménez et al. 2003). A continuación se describen los tipos de algoritmos más comunes.

Algoritmos secuenciales (Round Robin): son los más sencillos y consumen muy pocos recursos. Su funcionamiento se basa en realizar una selección secuencial y cíclica de los servidores reales. Son poco eficientes pues no tienen en cuenta el número de conexiones establecidas que tiene cada servidor (Martínez Jiménez et al. 2003).

Algoritmos basados en el menor número de conexiones (Least Connection): su funcionamiento se basa en el conocimiento que tiene el balanceador de carga del número de conexiones establecidas en cada servidor real. Con ese conocimiento, el balanceador de carga, a la hora de reenviar una nueva conexión, lo hace al servidor real que tenga menos conexiones activas (Martínez Jiménez et al. 2003).

Algoritmos basados en una distribución ponderada por pesos: permiten al administrador del sistema asignar un peso a cada servidor en función de su capacidad de procesamiento y el servicio presta. Se utilizan en combinación con los algoritmos secuenciales o con los basados en el menor número de conexiones (Martínez Jiménez et al. 2003).

Métodos de reenvío de paquetes

Los métodos de reenvío consisten en la forma en que el balanceador de carga reenvía los paquetes a los servidores reales y dependen en gran medida de la topología de red que interconecta los nodos del cluster. Si los servidores reales están ubicados en redes privadas, donde no pueden ser accedidos directamente por los clientes, lo más común es utilizar Traducciones de Direcciones de Red (NAT, por sus siglas en inglés). Cuando los servidores reales están ubicados en la misma subred que el balanceador de carga y pueden responder directamente a los clientes, el método Respuesta Directa del Servidor (DSR, por sus siglas en inglés) es el más común. En caso que los servidores se encuentren separados geográficamente y en redes diferentes se pueden utilizar técnicas de balanceo de carga global (Martínez Jiménez et al. 2003).

Traducciones de Direcciones de Red

Los primeros balanceador de carga se basaron en NAT, que es un método que se utiliza para conectar un dominio de direcciones privadas no registradas con un dominio externo de direcciones registradas globalmente (Srisuresh et al. 1999).

Específicamente *Destination* NAT (Srisuresh et al. 1999) es la variante de NAT que utilizan los balanceadores para este método de reenvío de paquetes. El método consiste en cambiar la dirección IP destino de la cabecera IP de los paquetes enviados por el cliente.

Los paquetes llegan al balanceador de carga con la VIP como dirección IP destino, este la sobrescribe utilizando la dirección IP del servidor real al cual los reenvía. Cuando el paquete retorna al cliente, el balanceador de carga restituye la VIP como dirección origen (Martínez Jiménez et al. 2003). En la figura 1.2 se muestra una descripción del proceso de traducción de direcciones IP que ocurre en el balanceador de carga.

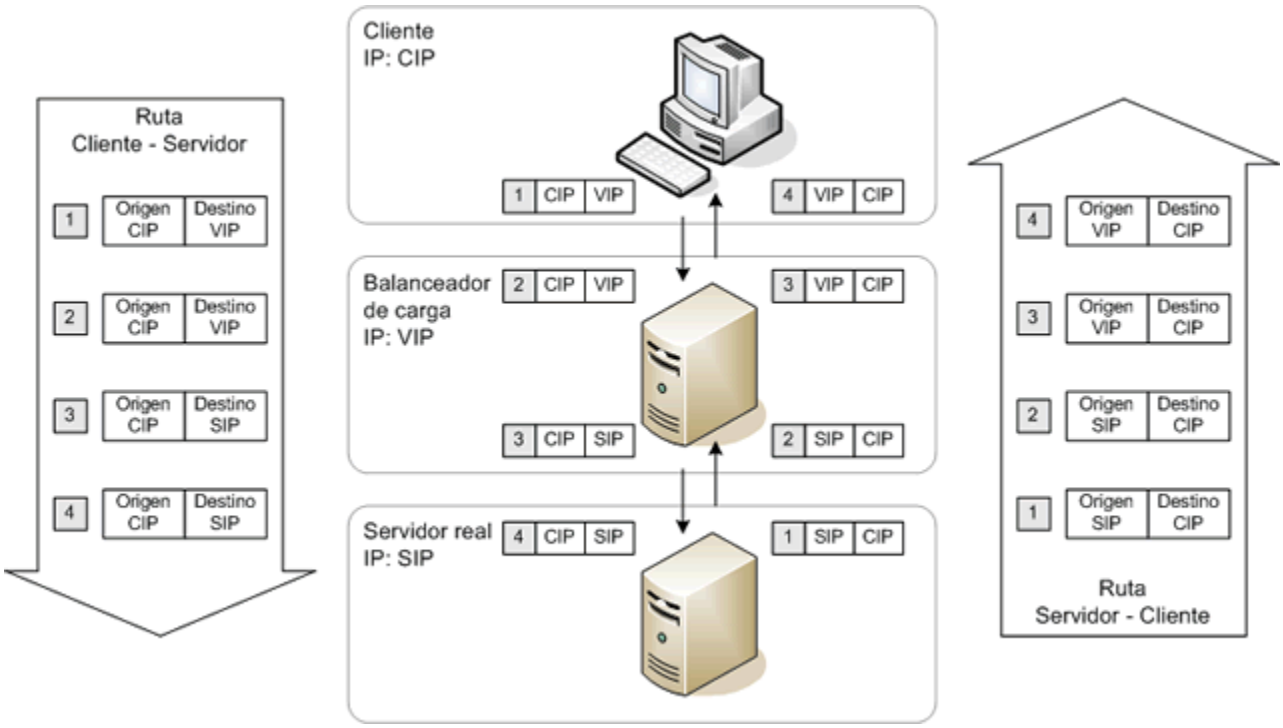

Figura 1.2. Traducción de direcciones de red destino.

Para que este método de reenvío de paquetes funcione correctamente, los servidores reales tienen que tener configurado como puerta de enlace por defecto, la dirección IP que utiliza el balanceador de carga para comunicarse con ellos.

Respuesta Directa del Servidor

DSR es una técnica que introducir importantes mejoras en cuanto a la eficiencia del balanceador de carga. Las mejoras se consiguen gracias a que el tráfico de respuesta, generado por los servidores reales, no tiene que pasar a través del balanceador de carga. Los paquetes de respuestas son transmitidos directamente desde los servidores reales hacia los clientes (Martínez Jiménez et al. 2003).

Utilizando este método el balanceador de carga no necesita sobrescribir las cabeceras IP de los paquetes, pues el reenvío se realiza a nivel de la capa de enlace de datos del modelo OSI (*Open Standard Interconection*) de la ISO (*International Organization for Standardization*). Para utilizar este método de reenvío de paquetes, tanto el balanceador de carga como los servidores reales, tienen que pertenecer al mismo dominio de *broadcast* (Martínez Jiménez et al. 2003).

Como el balanceador de carga no sobrescribe la dirección IP destino de los paquetes antes de reenviarlos, al llegar a los servidores reales tienen la VIP como dirección IP destino. Para que los servidores reales no rechacen los paquetes que reciben del balanceador de carga, es necesario que el servicio al que van dirigidos esté escuchando por la VIP. Para que en cada servidor real un servicio escuche por la VIP, se necesita un dispositivo de red que esté configurado con ella (Martínez Jiménez et al. 2003).

El protocolo de resolución de direcciones (ARP, por sus siglas en inglés) es utilizado normalmente para resolver la dirección de Control de Acceso al Medio (MAC, por sus siglas en inglés) asociada a una determinada dirección IP dentro de un mismo dominio de *broadcast*. En DSR, tanto el balanceador de carga como los servidores reales tienen un dispositivo de red configurado con la VIP. Si todos contestan a las peticiones ARP que realiza el enrutador, este podría enviar las conexiones de los clientes a un servidor real en lugar de al balanceador de carga. Para evitar este problema, los dispositivos de red que se configuren con la VIP en los servidores reales no pueden responder las peticiones ARP que realiza el enrutador (Martínez Jiménez et al. 2003).

Los servidores reales, además de disponer de un dispositivo configurado con la VIP, deben tener otro que les permita comunicarse con el balanceador de carga. La dirección MAC de estos dispositivos es la que utiliza el balanceador de carga para encapsular los paquetes antes de reenviarlos.

Cuando los servidores reales generan los paquetes de respuesta ponen como dirección IP destino la del cliente y como origen la VIP. De este modo se evita que el paquete tenga que pasar a través del balanceador de carga antes de ser enviado al cliente, como era necesario hacer en el método basado en NAT (Martínez Jiménez et al. 2003).

Persistencia de sesión

Existen servicios que intercambian datos a nivel de capa de aplicación del modelo OSI de la ISO, utilizando para ello el establecimiento de varias conexiones TCP (*Transmission Control Protocol*). Los servicios con esas características reciben el nombre de servicios transaccionales (Martínez Jiménez et al. 2003).

El balanceador de carga puede distribuir las conexiones provenientes de un mismo cliente entre varios servidores reales. Como resultado la información necesaria para completar un servicio puede estar distribuida parcialmente en más de un servidor real. Lo antes descrito trae como consecuencia que el correcto funcionamiento de un servicio transaccional se vea afectado por el comportamiento normal de un balanceador de carga.

La persistencia de sesión en el balanceo de carga garantiza que todas las conexiones provenientes de una misma dirección IP sean reenviadas a un mismo servidor real. Para hacerlo el balanceador de carga compara cada conexión nueva con las ya establecidas. Si encuentra alguna coincidencia, la nueva petición es reenviada al mismo servidor real que atiende o atendió la otra. En caso contrario el balanceador escoge un servidor real, según el algoritmo de distribución de carga que utilice y hacia él reenvía la nueva conexión (Martínez Jiménez et al. 2003).

El uso de persistencia de sesión no se recomienda para balancear servicios que son accedidos desde un número pequeño de direcciones IP, porque afecta el equilibrio en la distribución de carga.

Escalabilidad

La escalabilidad de un cluster de balanceo de carga viene dada por la facilidad de agregar servidores reales al cluster cuando aumenta la demanda del servicio que se balancea. Así se logra aumentar la capacidad de procesamiento y se evita una sobrecarga en los servidores reales que puede provocar el colapso del sistema completo. De manera análoga cuando disminuye la demanda del servicio se pueden retirar servidores reales para evitar una subutilización de los mismos (Martínez Jiménez et al. 2003).

Recuperación ante fallos en los servidores reales

Para evitar que la falla del servicio en alguno de los servidores reales afecte el funcionamiento del sistema, el balanceador debe tener la capacidad de conocer, en todo momento, el estado de todos sus servidores reales. Así cuando detecta que un servidor ha dejado de prestar servicio se puede reconfigurar para dejar de reenviarle conexiones (Martínez Jiménez et al. 2003).

Aunque las conexiones que los clientes tenían establecidas con el servidor que quedó fuera de servicio se pierden, se logra que la afectación al servicio sea mínima. De manera análoga, cuando el balanceador de carga detecta que un servidor que estaba de baja se encuentra listo para prestar servicio, puede reconfigurarse y comenzar a reenviarle conexiones (Martínez Jiménez et al. 2003).

Alta disponibilidad

En un cluster de balanceo de carga no solo basta con garantizar la recuperación ante fallos en los servidores reales, pues el balanceador de carga constituye un punto único de fallo. Existen técnicas para implementar alta disponibilidad que garantizan que el cluster siga prestando servicios aún cuando falla el balanceador de carga.

La alta disponibilidad se basa en la redundancia de recursos y para obtener un cluster que la implemente se necesitan dos balanceadores carga. Esta se puede logar utilizando el método activo/pasivo o activo/activo. Con el primero un balanceador de carga presta servicios y el otro se encuentra listo para tomar su lugar si este falla. En el segundo, ambos balanceadores se encuentran prestando servicio de forma activa y listos para asumir el trabajo del otro en caso de que falle.

El uso de esta técnica no evita la caída del servicio cuando falla el balanceador de carga, solo logra que la recuperación sea casi instantánea y sin la intervención del administrador del sistema.

1.3. Información persistente de aplicaciones web en un cluster de balanceo de carga

El o los métodos que utiliza una aplicación web para almacenar su información persistente del lado del servidor, influye en el diseño de un cluster de balanceo de carga que se realice para brindar este servicio.
Existen tres formas generales que puede utilizar una aplicación web para almacenar su información persistente del lado del servidor:

- En bases de datos.
- En memoria RAM.
- En el sistema de ficheros.

A continuación se analizan las principales variantes que se utilizan en el diseño de un cluster de balanceo de carga para aplicaciones web que usan alguno de los métodos de almacenamiento antes mencionados.

Información persistente en bases de datos

Si la información persistente que genera o utiliza la aplicación web se almacene en bases de datos, el diseño del clúster de balanceo de carga se realiza de la forma más básica. No es necesario introducir variantes debido a que independientemente de que servidor real genere o modifique la información, esta siempre va a estar disponible y actualizada para el resto.

Información persistente en memoria RAM

Cuando la información persiste en la memoria RAM el diseño del cluster debe garantizar que todas las conexiones que realiza el cliente sean atendidas por el mismo servidor real. Para lograrlo es necesario utilizar los mecanismos de persistencia de sesión basados en dirección IP origen que implementan los balanceadores de carga. Se debe tener presente que este diseño puede afectar el equilibrio en el ba-

lanceo de carga, si el acceso de los clientes a la aplicación se realiza desde una dirección IP o un pequeño número de ellas.

Información persistente en el sistema de ficheros

Si la información persistente que genera o utiliza la aplicación web es almacenada en el sistema de ficheros, para realizar el diseño es necesario analizar su tiempo de vida útil. En el caso que la información tenga una larga vida útil, lo más común es crear un sistema de archivos distribuido para su almacenamiento. Así se garantiza que siempre este actualizada y disponible para todos los servidores web que componen el cluster. Este diseño tiene como inconveniente que puede necesitar más servidores para su implementación y afectar ligeramente el rendimiento del sistema.

Cuando la información que se almacena tiene un tiempo de vida relativamente corto, se puede tratar como un almacenamiento en memoria RAM y realizar el diseño teniendo en cuanta las mismas consideraciones. Si se desea evitar el riesgo de afectar el balanceo de carga, en caso que el acceso a la aplicación se realice desde una o un reducido número de direcciones IP, se puede utilizar un sistema de archivos distribuido para su almacenamiento.

1.4. Balanceadores de carga desarrollados sobre software libre

Para implementar un cluster de balanceo de carga es necesario utilizar algún software que realice esta función. En la actualidad existen varios balanceadores de carga, algunos propietarios y otros libres. Los que se describen a continuación se clasifican como libres.

Kernel TCP Virtual Server

Kernel TCP Virtual Server (KTCPVS) es un balanceador de carga desarrollado por el proyecto Linux Virtual Server. Este implementa balanceo de carga dentro del kernel de Linux a nivel de la capa de aplicación, del modelo OSI de la ISO (Linux Virtual Server Project 2006).

Al KTCPVS ser un balanceador de carga que trabaja a nivel de capa de aplicación, brinda una solución poco escalable. La poca escalabilidad es producto a la sobrecarga que genera en el servidor sobre el que se ejecuta. El tener conocimiento del contenido de los paquetes antes de reenviarlos a los

servidores reales lo convierte en un balanceador de carga muy flexible (Linux Virtual Server Project 2006).

El desarrollo de KTCPVS comenzó en el año 2000, pero hace más de tres años que este *software* no ha sido actualizado. Todavía no se ha liberado una versión estable de este balanceador, por lo que no se recomienda su uso en entornos de producción.

Crossroads

Crossroads es un balanceador de carga libre y de código abierto que trabaja a nivel de capa de aplicación del modelo OSI de la ISO. Fue diseñado para balancear carga en servicios TCP como: HTTP, HTTPS, SSH, SMTP y DNS principalmente (Kubat 2008b).

Su principio de funcionamiento es el de un balanceador de carga normal, pero además incluye facilidades para garantizar recuperación ante fallos en los servidores reales. También puede ser utilizado en sistemas sin discos (Kubat 2008a).

Crossroads no es un balanceador de carga muy fiable y constituye un punto único de fallo para el sistema completo (Kubat 2008a). La primera versión de este *software* se liberó en el mes de enero del año 2007 pero todavía no es muy utilizado.

HAProxy

HAProxy es una solución libre que ofrece servicios de alta disponibilidad, balanceo de carga y proxy para aplicaciones basadas en el protocolo HTTP. Se diseño originalmente para servir de proxy a aplicaciones web que requieren persistencia a nivel de capa de aplicación del modelo OSI de la ISO. En versiones posteriores se le añadió la facilidad de balanceo de carga a nivel de capa aplicación (Tarreau 2008).

Actualmente HAProxy no soporta SSL (Secure Socket Layer), ni Keep-Alive. El desarrollador del *software* planea implementarlo en un futuro próximo (Tarreau 2008).

Linux Virtual Server

El proyecto Linux Virtual Server (LVS) trabaja por obtener un Servidor Virtual, cluster, de alta disponibilidad y escalabilidad sobre el sistema operativo Linux. La arquitectura del Servidor Virtual es completamente transparente a los usuarios finales, los cuales utilizan el cluster como si fuera un servidor de alto rendimiento (Linux Virtual Server Project 2005b).

Un cluster de tipo LVS se compone por dos tipos de servidores: los balanceadores de carga y los servidores reales. La conexión entre los servidores puede ser a través de una red LAN de alta velocidad o de una red WAN si están distribuidos geográficamente (Linux Virtual Server Project 2005b).

El *software* que utiliza LVS para su implementación es IPVS (IP Virtual Server), este es un balanceador de carga que trabaja en la capa de trasporte del modelo OSI de la ISO y actualmente es un módulo del kernel de Linux (Martínez Jiménez et al. 2003).

IPVS incluye varios algoritmos distintos para distribuir la carga entre los servidores reales. A continuación se describen los más utilizados.

- Round-Robin (rr)

  Para utilizar este algoritmo el balanceador de carga crea una lista circular con los servidores reales del cluster y reenvía cada conexión al próximo servidor de la lista (Linux Virtual Server Project 2005a).

- Weighted Round-Robin (wrr)

  Este algoritmo es una mejora del Round-Robin que se basa en la distribución ponderada por pesos (Linux Virtual Server Project 2005a).

- Least-Connection (lc)

  Es un algoritmo de naturaleza dinámica que dirige las conexiones al servidor real que tenga el menor número de estas activas. Está diseñado para entornos donde cada conexión puede generar una carga de trabajo diferente en el servidor que la procesa. Para utilizar este algoritmo el balanceador de carga necesita llevar el control del número de conexiones activas que tienen los servidores reales en cada momento (Linux Virtual Server Project 2005a).

- Weighted Least-Connection (wlc)

  El algoritmo Weighted Least-Connection es un perfeccionamiento del Least-Connection el cual se basa en la distribución ponderada por pesos (Linux Virtual Server Project 2005a).

El balanceador de carga IPVS garantiza la persistencia de sesión basada en direcciones IP origen. Para lograrlo mantiene el control sobre el inicio y finalización de las conexiones TCP que distribuye a los servidores reales (Martínez Jiménez et al. 2003).

El balanceador de carga IPVS soporta tres métodos de reenvío de paquetes. Los dos primeros son LVS-NAT y LVS-DR, que utilizan Traducción de Direcciones de Red y Respuesta Directa del Servidor respectivamente. El tercero, LVS-TUN, se utiliza para balanceo de carga global y se basa en LVS-DR, pero está diseñado trabajar con servidores reales que pueden estar separados geográficamente (Martínez Jiménez et al. 2003).

El proyecto Linux Virtual Server, paralelamente al balanceador de carga IPVS, desarrolla una herramienta para su administración llamada Ipvsadm. Esta herramienta es que se utiliza para la configuración y gestión del balanceador de carga IPVS.

IPVS es el balanceador de carga que se escogió para implementar el cluster de balanceo de carga que constituye la base de la propuesta de solución que se obtendrá como resultado de esta investigación.

1.5. Complemento de software necesario para el cluster

Para que un cluster de balanceador de carga implementado con IPVS incluya recuperación ante fallos en los servidores reales y alta disponibilidad es necesario utilizar otros tipos de software. Entre las propuestas que se realizan en el sitio web del proyecto LVS para logar este objetivo, la más utilizada es la combinación Ldirectord + Heartbeat. A continuación se describen las características de cada uno de ellos.

Linux Director Daemon

Linux Director Daemon (Ldirectord) es un *software* programado en Perl para configurar dinámicamente al IPVS mediante el Ipvsadm. Este monitoriza los servicios ofrecidos por una granja de servidores reales balanceados mediante LVS para garantizar su disponibilidad. Tiene la capacidad de añadir o borrar entradas en la tabla de sesiones del balanceador de carga al detectar un servidor de alta o fuera de servicio respectivamente. De esta forma logra mantener actualizadas las tablas sesiones que definen el comportamiento del balanceador de carga IPVS. Así, cuando en uno de los servidores reales deja de funcionar alguno de los servicios prestados por el cluster, el Ldirectord hará que el balanceador deje de reenviarle conexiones a este (Horman et al. 2006).

Heartbeat

Heartbeat es un *software* desarrollado para mantener alta disponibilidad en servicios que se ejecuten sobre el sistema operativo Linux. Se utiliza para la implementación de un cluster de alta disponibilidad IP de tipo activo/pasivo (Linux HA Project 2008).

Para implementar de un cluster de alta disponibilidad utilizando Heartbeat se requiere un mínimo de dos servidores capacitados para prestar el servicio. Uno de los servidores, el primario, presta el servicio de manera activa y el otro, el secundario, se mantiene supervisándolo. Así, si el servidor primario deja de prestar el servicio, el servidor secundario lo detecta, toma su lugar y sigue prestándolo (Martínez Jiménez et al. 2003).

La supervisión se basa en el envío y recepción de “latidos” entre los demonios Heartbeat que corren en ambos servidores. Cuando el servidor secundario deja de escuchar los “latidos” del servidor primario por un determinado periodo de tiempo, supone que este dejo de funcionar y toma su lugar. Tan pronto como el servidor secundario vuelve a escuchar los “latidos” del servidor primario, detiene los servicios que está ofreciendo para que el otro tome el lugar que le corresponde (Martínez Jiménez et al. 2003).

Los servidores primario y secundario deben mantener una comunicación directa. Actualmente Heartbeat soporta conexiones por puerto serie e interfaz de redes de tipo Ethernet. Para evitar que un

error en la comunicación haga que ambos servidores intenten prestar el servicio es recomendable el uso de conexiones redundantes (Linux HA Project 2008).

1.6. Software para sistema de archivos distribuido

Un sistema de archivos distribuido almacena ficheros en una o más computadoras denominadas servidores y los hace accesibles a otras, denominadas clientes, quienes manipulan los ficheros como si fueran locales. El software existente para implementar un sistema de archivos distribuido es muy variado. A continuación se describen dos de los más utilizados.

Network File System

Network File System (Sistema de archivos de red, NFS) es un protocolo de nivel de aplicación, según el Modelo OSI de la ISO. Se utiliza para la implementación de sistemas de archivos distribuidos en un entorno de red de computadoras de área local. Posibilita que distintos ordenadores conectados a una misma red accedan a ficheros remotos como si se tratara de ficheros locales (Sandberg et al. 1985).

NFS fue desarrollado en 1984 por Sun Microsystems, con el objetivo de independizar de la máquina, el sistema operativo y el protocolo de transporte (Sandberg et al. 1985). Actualmente el protocolo NFS está incluido por defecto en los sistemas operativos UNIX y las distribuciones Linux.

El sistema de archivos distribuido NFS se divide en dos partes principales: el servidor y los clientes. En el servidor se centraliza toda la información para que pueda ser accedida de forma remota por los clientes, evitando la replicación datos (Sandberg et al. 1985).

Para garantizar la integridad de los datos en un sistema de archivos distribuido NFS, todas las operaciones que se realizan sobre los ficheros son síncronas. Además, cuando se realiza una solicitud de escritura, el servidor escribirá físicamente los datos en el disco, y si es necesario, actualizará la estructura de directorios, antes de devolver una respuesta al cliente (Sandberg et al. 1985).

Coda

Coda es un sistema de archivos distribuido desarrollado como un proyecto de investigación en la Universidad Carnegie Mellon. Comenzó desde año 1987 bajo la dirección de M. Satyanarayanan. Este sistema de archivos distribuido desciende directamente de una antigua versión de AFS (Andrew File System) y ofrece muchas características similares (Coda Project 2008).

Algunas de las principales características del sistema de archivos distribuido Coda son:

- Puede funcionar sin conexión.
- Es software libre.
- Obtiene alto rendimiento gracias al caché persistente en los clientes.
- Permite la replicación de servidores.
- Implementa un modelo de seguridad para autenticación, cifrado y control de acceso.

Actualmente se está realizando un importante esfuerzo para mejorar Coda. Este se centra principalmente en mejorar las prestaciones y la fiabilidad, portarlo a distintas plataformas y ampliar la documentación (Coda Project 2008).

1.7. Pruebas para validar el despliegue de las aplicaciones web

Cuando se desea aumentar la capacidad de procesamiento en los servidores web sobre los que se ejecutan las aplicaciones de tipo web, generalmente se introducen cambios en la arquitectura de despliegue. Si la solución utilizada para lograr este objetivo se basan en el uso de cluster de balanceo de carga, los cambios son muy significativos y es necesario realizar pruebas que permitan validar la nueva arquitectura de despliegue.

Descripción de las pruebas

Las pruebas necesarias para validar una arquitectura de despliegue de una aplicación web pueden ser clasificadas según el objetivo que se persigue al realizarlas:

- Pruebas de configuración.

  Este tipo de pruebas se realiza para garantizar que la aplicación funcione correctamente sobre diferentes configuraciones de hardware y/o software (Rational Unified Process 2003). Durante su realización tiende a ocurrir un ciclo de perfeccionamiento en el diseño de la solución.

  Las pruebas de configuración constituyen la base para validar una nueva arquitectura de despliegue. Los resultados que se obtienen al realizarlas permiten decidir si se hacen pruebas de carga o no.

- Pruebas de rendimiento.

  Este tipo de pruebas se realiza para determinar o validar la velocidad, escalabilidad y la estabilidad de la aplicación bajo prueba. El rendimiento se refiere principalmente a la información relativa a los tiempos de respuestas y la utilización de recursos (Meier et al. 2007b).

- Pruebas de stress.

  Este tipo de pruebas constituye una subcategoría de las pruebas de rendimiento. Son diseñadas para evaluar como la aplicación responde bajo condiciones anormales como una sobrecarga extrema (Meier et al. 2007b).

- Pruebas de carga.

  Son una subcategoría de las pruebas de rendimiento. Tienen como objetivo analizar el comportamiento de una aplicación al ser sometido a los diferentes niveles de carga previstos para las operaciones de producción (Meier et al. 2007b). Permiten definir los límites operacionales de una configuración para los diferentes niveles de carga.

- Pruebas de benchmark.

  Este tipo de prueba permite comparar el rendimiento de un nuevo o desconocido objeto de pruebas respecto al de un sistema o referencia conocida (Rational Unified Process 2003). Es muy utilizado para definir entre diferentes arquitecturas de despliegues o configuraciones específicas de estas cual es la mejor para dar solución a un determinado problema.

Para realizar las pruebas de carga, benchmark y stress es necesario utilizar herramientas que permitan generar carga.

Medición del rendimiento

Para poder evaluar el rendimiento de un sistema, es necesario realizar mediciones del comportamiento de algunas de sus características y expresarlas utilizando alguna escala común. Cuando se realizan mediciones, se puede recoger información referente al funcionamiento de todo el sistema y/o de los servidores que lo conforman.

Las mediciones para recolectar información del funcionamiento de todo un sistema se basan principalmente en la capacidad de este para responder a las peticiones que realizan los clientes (Meier et al. 2007a). Las más comunes son:

- Número de peticiones respondidas por segundo.
- Número de peticiones completadas.
- % de peticiones que fallidas.
- Tiempo de respuesta promedio del sistema.

Cuando se trata de analizar el comportamiento de los servidores que forman parte del sistema las mediciones se enfocan en el consumo de los recursos (Meier et al. 2007a). Las mediciones que normalmente se realizan en los servidores son:

- % de uso del CPU.
- % de consumo de memoria RAM.
- Tasa de transferencia a disco.
- % de uso de la red.

1.8. Otros tipos de software y herramientas que se utilizan

Tanto para el proceso de pruebas como para la gestión del monitoreo y la seguridad del cluster una vez que ha sido desplegado es necesario utilizar software extra al que ya se menciona anteriormente. Todo este software se describe a continuación.

JMeter

JMeter es una aplicación de escritorio desarrollada sobre Java. Originalmente fue diseñada para realizar pruebas de carga y medir el rendimiento de aplicaciones web. Su uso se ha extendido a otros tipos de pruebas(Apache Software Foundation 2008).

Actualmente el JMeter se puede utilizar para probar el rendimiento en recursos estáticos y dinámicos. Con esta herramienta se puede generar una pesada carga a servidores web, ftp y de bases de datos, para analizar su comportamiento bajo diferentes niveles de carga (Apache Software Foundation 2008).

Sysstat

Sysstat es un producto libre y de código abierto que contiene un grupo de herramientas para monitorizar el rendimiento y la actividad en un servidor. Con ellas se puede recolectar información y generar reportes sobre el consumo de recursos como: CPU, memoria RAM y red (Godard 2008).

Nagios

Nagios es una herramienta de monitoreo para fallas en la red, servidores y servicios y generar alarmas cuando las detectas. Ha sido diseñado para ejecutarse sobre el sistema operativo Linux. Su principio de funcionamiento es el chequeo intermitente de los recursos que monitorea. Genera reportes sobre el estado actual e histórico de los recursos, que pueden ser consultados a través de un navegador web (Nagios Enterprises 2008).

Net-SNMP

SNMP es un protocolo de capa de aplicación del modelo OSI de la ISO que facilita el intercambio de información de administración entre dispositivos de red. Forma parte de la familia de protocolos TCP/IP (Net-SNMP Project 2007).

Net-SNMP es un conjunto de herramientas, que utilizan el protocolo SNMP, para la monitorización y administración de dispositivos de red. Estas son clasificadas en dos grupos: los agentes, demonios que reciben, procesan y responden las peticiones SNMP (Debian Project 2008d) y los clientes,

aplicaciones de línea de comando que realizan peticiones SNMP a los agentes (Debian Project 2008c).

MRTG

MRTG es una herramienta utilizada principalmente para supervisar la carga de tráfico de red en este tipo de dispositivos. La recolección de información la realiza generalmente utilizando el protocolo SNMP. Los reportes los genera en imágenes, que proporcionan una representación visual del tráfico en los dispositivos monitorizados y los visualiza a través de páginas en formato HTML (Oetiker 2008).

Además de la vigilancia a través de SNMP, MRTG también pueden generar reportes basados en los resultados que ofrecen otras herramientas, *scripts* principalmente. De esta forma se puede monitorizar el comportamiento de otros recursos como CPU y memoria RAM (Oetiker 2008).

RRDtool

RRDtool (*Round Robin Database Tool*) es un sistema para almacenar y mostrar series cronológicas de datos. Los datos se almacenan en RRDs (Round Robin Databases, bases de datos circulares). Estas son muy compactas y no se amplían con el tiempo (Debian Project 2008b). A menudo es utilizado por otras herramientas, como MRTG, para almacenar datos y después utilizarlos para la generación de reportes.

Iptables

El software Iptables proporciona un framework al kernel de Linux para realizar filtrado de paquetes y traducción de direcciones de red y puertos. Se utiliza principalmente para implementar cortafuegos, servidores proxy transparentes, enrutamiento avanzado y control de tráfico de red (Debian Project 2008a).

1.9. Conclusiones

Después de realizar un estudio de las técnicas de cluster de balanceo de carga, el *software* que se utiliza para su implementación y las tipos de pruebas para validar una propuesta de solución basada en este tipo de cluster, se concluyó que:

1. La implementación del cluster de balanceo de carga que constituye la base de la propuesta de solución se realizará utilizando la combinación del balanceador de carga IPVS con el *software* Ldirectod y Heartbeat.
2. Es necesario realizar pruebas de configuración y carga para validar la propuesta de solución.
3. Las métricas que se utilizaran son: tiempo de respuesta para medir el rendimiento del sistema, % de uso de CPU, % de consumo de memoria RAM y % de uso de la red para el rendimiento de los servidores.

CAPÍTULO 2. ESTUDIO DE LAS APLICACIONES Y SELECCIÓN DE LA MUESTRA

Este capítulo está dedicado al estudio de las aplicaciones médicas de tipo web que desarrollan la Facultad 7 de la UCI y/o Softel. El objetivo que se persigue es seleccionar, de entre todas las aplicaciones, la muestra que será utiliza durante la investigación. También se realiza una descripción detallada de las aplicaciones de muestra y del entorno donde están desplegadas.

2.1. Características generales de las aplicaciones

El diseño de una solución basada en cluster de balanceo de carga es específico para cada aplicación. Por ese motivo se decidió indagar sobre las características de las aplicaciones médicas de tipo web que se desarrollan o han sido desarrolladas por la Facultad 7 de la UCI y/o por la empresa productora de software Softel.

La indagación se realizó para poder identificar todas las aplicaciones médicas de tipo web, que se desarrollan o han sido desarrolladas sobre software libre, su explotación sea realiza de forma centralizada y además son accedidas por un gran número de usuarios de manera concurrente.

Para conocer que aplicaciones cumplen con los requisitos antes mencionados, se decidió realizar entrevistas. Los entrevistados fueron los diseñadotes, arquitectos o jefes de proyectos de la Facultad 7 de la UCI y la empresa Softel. Como resultado de las entrevistas realizadas se pudo identificar las aplicaciones que se detallan en la tabla 2.1.

Tabla 2.1. Aplicaciones médicas de tipo web identificadas.

Nombre de la aplicación	Tipo	Despliegue	Niveles físicos del despliegue	Servidor de aplicación
Registro de problemas de salud de APS	web	Centralizado	3	Apache + PHP
Registro de ubicación	web	Centralizado	3	Apache + PHP
Registro de clasificación internacional de enfermedades y problemas relacionados de salud	web	Centralizado	3	Apache + PHP

Nombre de la aplicación	Tipo	Despliegue	Niveles físicos del despliegue	Servidor de aplicación
Registro de servicios médicos	web	Centralizado	3	Apache + PHP
Registro de estudiantes	web	Centralizado	3	Apache + PHP
Registro de áreas de salud	web	Centralizado	3	Apache + PHP
Registro de enfermedades de declaración obligatoria (*)	web	Centralizado	3	Apache + PHP
Registro de localidades	web	Centralizado	3	Apache + PHP
Registro de equipos médicos	web	Centralizado	3	Apache + PHP
Registro de equipos no médicos	web	Centralizado	3	Apache + PHP
Registro del ciudadano	web	Centralizado	3	Apache + PHP
Registro de personal de la salud	web	Centralizado	3	Apache + PHP
Registro de unidades de salud (*)	web	Centralizado	3	Apache + PHP
Registro de población	web	Centralizado	3	Apache + PHP
Registro de fallecidos (*)	web	Centralizado	3	Apache + PHP
Registro de partos y nacimientos (*)	web	Centralizado	3	Apache + PHP
Registro de actividades diarias (*)	web	Centralizado	3	Apache + PHP
Registro de vacunación (*)	web	Centralizado	3	Apache + PHP
Registro de indicadores y conductas de la atención primaria (*)	web	Centralizado	3	Apache + PHP
Versión en PHP del componente de seguridad de SISalud	web	Centralizado	3	Apache + PHP
Balance Material (**)	web	Centralizado	2	Apache + PHP
Sistema de información estadística complementaria de salud (**)	web	Centralizado	2	Apache + PHP

Nombre de la aplicación	Tipo	Despliegue	Niveles físicos del despliegue	Servidor de aplicación
Sistema de gestión de información del proceso de formación de recursos humanos en salud (**)	web	Centralizado	2	Apache + PHP
Registro centralizado de donantes	web	Centralizado	3	Apache-TomCat
Versión en Java del componente de seguridad de SISalud	web	Centralizado	3	Apache-TomCat

(*) Aplicaciones en desarrollo
(**) Aplicaciones parcialmente desplegadas

La aplicación Registro centralizado de donantes no gestiona altos niveles de concurrencia. Pero como tiene las características de un grupo de aplicaciones que serán desarrolladas en un futuro próximo, que si van a ser accedidas por un gran numero de usuarios de manera concurrente, se decidió incluirla entre las aplicaciones que requieren este tipo de solución.

2.2. Estratificación de las aplicaciones y selección de las aplicaciones de muestra

Para facilitar el estudio, las aplicaciones médicas de tipo web que se identificaron en el epígrafe 2.1 fueron agrupadas. La agrupación se realizó teniendo en cuenta como principales características el número de niveles del despliegue y el servidor de aplicación que utilizan. Como resultado se obtuvieron tres grupos de aplicaciones médicas.

En el primer grupo se encuentran todas las aplicaciones médicas de tipo web que se pueden desplegar en tres niveles físicos y utilizan el servidor de aplicación Apache + PHP. El segundo grupo se conformó con las aplicaciones que pueden ser desplegadas en dos niveles físicos y utilizan el servidor de aplicación Apache + PHP. En el tercer y último están las aplicaciones que pueden ser desplegadas en tres niveles físicos y utilizan el servidor de aplicación Apache-Tomcat. La gráfica de la figura 2.1 muestra el resultado de la estratificación de las aplicaciones.

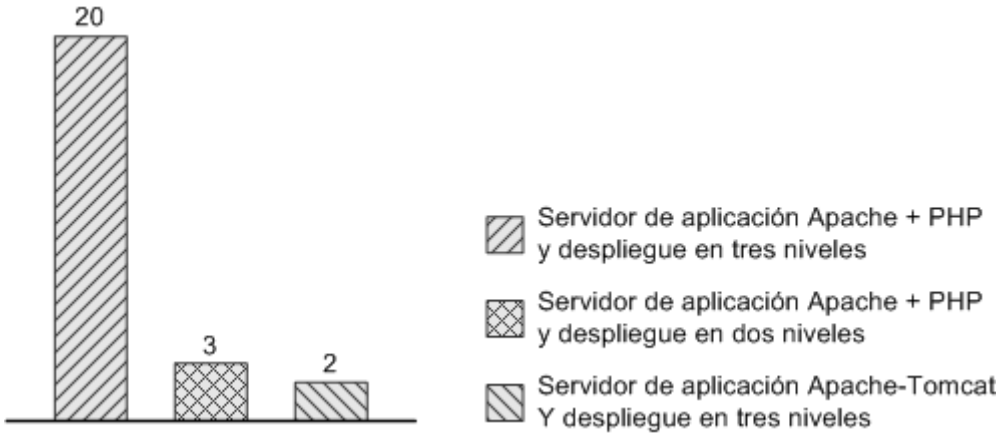

Figura 2.1. Número de aplicaciones por grupos.

Selección de las aplicaciones representativas

Debido a la cantidad de aplicaciones y al hecho de que no todas están disponibles para la realización de pruebas, se decidió seleccionar una aplicación representativa de cada grupo. El propósito es diseñar una o varias propuestas de solución para cada una de las aplicaciones representativas. Como todas las aplicaciones pertenecientes a un mismo grupo tienen las mismas características y se han desarrollado siguiendo las mismas normas, una solución valida para una lo debe ser para el resto. Las aplicaciones seleccionadas como representativas de cada grupo se especifican en la tabla 2.2.

Tabla 2.2. Aplicaciones representativas por cada grupo.

Grupo	Aplicación representativa
Servidor de aplicación Apache + PHP y despliegue en tres niveles	Registro de personal de la salud
Servidor de aplicación Apache + PHP y despliegue en dos niveles	Balance Material
Servidor de aplicación Apache-TomCat y despliegue en tres niveles	Registro centralizado de donantes

2.3. Descripción de las aplicaciones de muestra

Después de haber seleccionado las aplicaciones representativas se realizó una segunda entrevista a los diseñadores, arquitectos o jefes de proyectos donde se desarrollaron estas. El objetivo de esta segunda entrevista era obtener más información y poder hacer una descripción más detallada de las aplicaciones, para poder diseñar una propuesta de solución que permita su correcto funcionamiento.

Descripción de la aplicación de muestra Registro de Personal de la Salud

El Registro de Personal de la Salud (RPS) es una aplicación web desarrollada por la empresa productora de software Softel para el Ministerio de Salud Pública (MINSAP), con el objetivo de mantener un registro actualizado de todo el personal que trabaja en la salud. Actualmente está orientado a registrar toda la información relativa a profesionales médicos y no médicos (Softel 2006).

Esta aplicación fue creada para ser utilizada de forma centralizada, lo que permite que sea accedida desde todas las Unidades de Salud del país. Los requerimientos de software necesarios para su despliegue son: sistema operativo Linux, actualmente se utiliza la distribución White Box Enterprise Linux 3.0, el servidor web Apache en la versión 1.3.33, el lenguaje de programación PHP en la versión 4.3.10 integrado con el Sablotron en la versión 1.0, el servidor de bases de datos MySQL en la versión 4.0.13 y el Turck MMCache en la versión 2.4.6 para acelerar el procesamiento de código PHP.

El RPS forma parte de un grupo de registros primarios que componen el Registro Informatizado de Salud (RIS), los cuales tienen como objetivo logar que se codifique de manera uniforme, para que el resto de las aplicaciones en desarrollo y otras que serán desplegadas a corto, mediano y largo plazo puedan enviar información a los niveles superiores de manera única y la misma pueda ser procesada adecuadamente (Softel 2006).

Las aplicaciones del RIS, y por ende el RPS, está concebido completamente sobre una arquitectura basada en componentes y orientada a servicios. Utiliza PLASER (PLAtaforma de SERvicios) que es un *framework* que facilita la programación y homogeneidad de los componentes (Softel 2006).

La aplicación utiliza una arquitectura de tres capas distribuidas en tres niveles. Según este modelo la solución se encuentra segmentada en tres capas lógicas: presentación, negocio y datos y cada una de

estas se despliega en un servidor independiente (Softel 2006). En la figura 2.2 se muestra dicha arquitectura de despliegue.

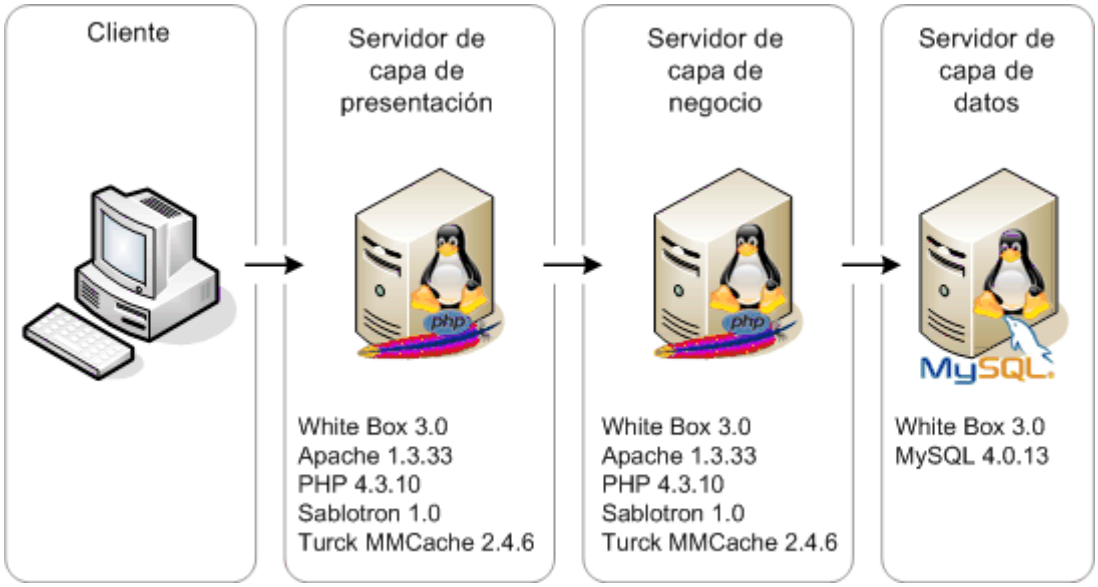

Figura 2.2. Arquitectura de despliegue de la aplicación Registro de Personal de la Salud.

Los usuarios solo pueden acceder a la capa de presentación. Esta contiene la interfaz de usuario, el negocio de validación de la entrada de datos por formularios y otras funcionalidades para garantizar la usabilidad del sistema (Sánchez Rodríguez et al. 2007). En esta capa se gestiona todo lo relacionado con la autentificación y autorización de los usuarios, para lo que se utiliza los servicios web que brinda el componente de seguridad de SISalud SAAA para este grupo de aplicaciones. El proceso de autentificación genera información de estado que el usuario necesita para interactuar con la aplicación, la cual se almacena del lado del servidor en los ficheros de sesión que genera el PHP.

La capa de negocio contiene los servicios que dan cumplimiento a los requisitos funcionales del sistema (Sánchez Rodríguez et al. 2007), estos están implementados como servicios web que utiliza la capa de presentación y otras aplicaciones. Entre los requisitos se encuentra la generación de reportes en formato PDF y EXCEL, los cuales se almacenan temporalmente en el sistema de ficheros del servidor sobre el que se ejecuta dicha capa.

La capa de datos contiene las bases de datos relacionales del sistema y solo es accedida desde la de negocio.

Descripción de la aplicación de muestra Balance Material

Balance Material es una aplicación web desarrollada por la Facultad 7 de la UCI para el Ministerio de Salud Pública (MINSAP), con el objetivo de automatizar el proceso de planificación de los pedidos de materiales gastables de uso médico en hospitales, policlínicos y clínicas estomatológicas.

Esta aplicación, que será utilizada de forma centralizada, se despliega sobre la distribución Debian GNU/Linux 4.0 del sistema operativo Linux, necesita el servidor web Apache en la versión 2.2.3, el lenguaje de programación PHP en la versión 5.2.0, integrado al motor de plantillas Smarty en la versión 2.6.14 y el servidor de bases de datos MySQL en la versión 5.0.32.

La arquitectura de la aplicación Balance Material se concibió según el patrón Modelo Vista Controlador. Utilizando este patrón se logra separar la aplicación en tres capas: presentación para la interfaz de usuario, lógica de negocio para implementar los requisitos funcionales del sistema y datos para la base de datos relacional del sistema.

El despliegue de la aplicación se realiza en dos niveles, el de aplicación para las capas de presentación y lógica de negocio y el de datos para la base de datos. Cada nivel se puede desplegar sobre un servidor independiente. En la figura 2.3 se muestra la arquitectura de despliegue para esta aplicación.

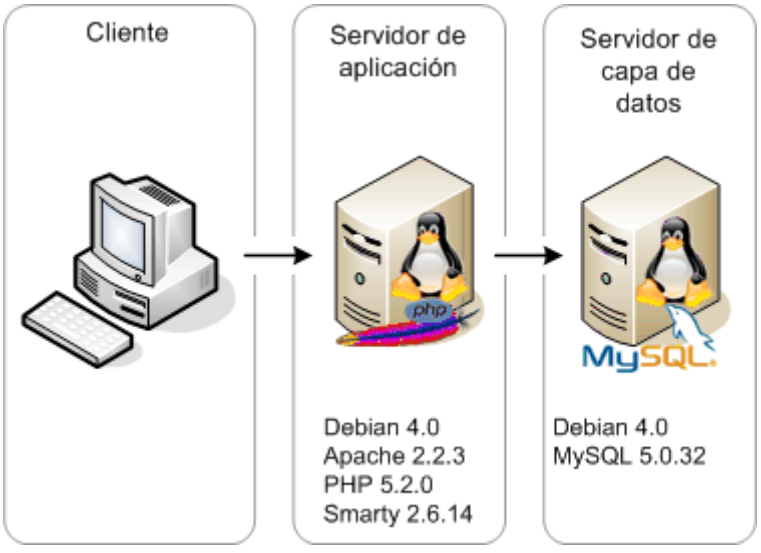

Figura 2.3. Arquitectura de despliegue de la aplicación Balance Material.

La capa de aplicación contiene la interfaz de usuario, el negocio de validación de la entrada de datos por formularios y otras funcionalidades para garantizar la usabilidad del sistema. Aquí también se gestiona la autenticación y autorización de los usuarios utilizando los servicios web que brinda el componente de seguridad de SISalud. El proceso de autenticación genera información de estado que el usuario necesita para interactuar con la aplicación. Esta es guardada en los ficheros de sesión que genera el PHP, los que se almacenan en el disco duro del servidor.

Descripción de la aplicación Registro Centralizado de Donantes

El Registro Centralizado de Donantes (RCD) es una aplicación web desarrollada por la empresa productora de software Softel para el MINSAP. El propósito principal de esta aplicación es mantener un registro centralizado con toda la información referente a las donaciones de sangre que se realizan en el país.

Esta aplicación se diseño para utilizarse de forma centralizada y ser accedida desde todos los bancos de sangre del país. La aplicación corre sobre la distribución Debian GNU/Linux 4.0 del sistema operativo Linux, utiliza el servidor de aplicación Apache Tomcat en su versión 5.5.23 y el servidor de bases de datos MySQL en la versión 5.0.32.

La arquitectura de la aplicación RCD está basada en componentes y orientada a servicios, se compone por tres capas lógicas: presentación, negocio y datos y cada una de estas se despliega en un servidor independiente, como se mostró en la figura 2.4.

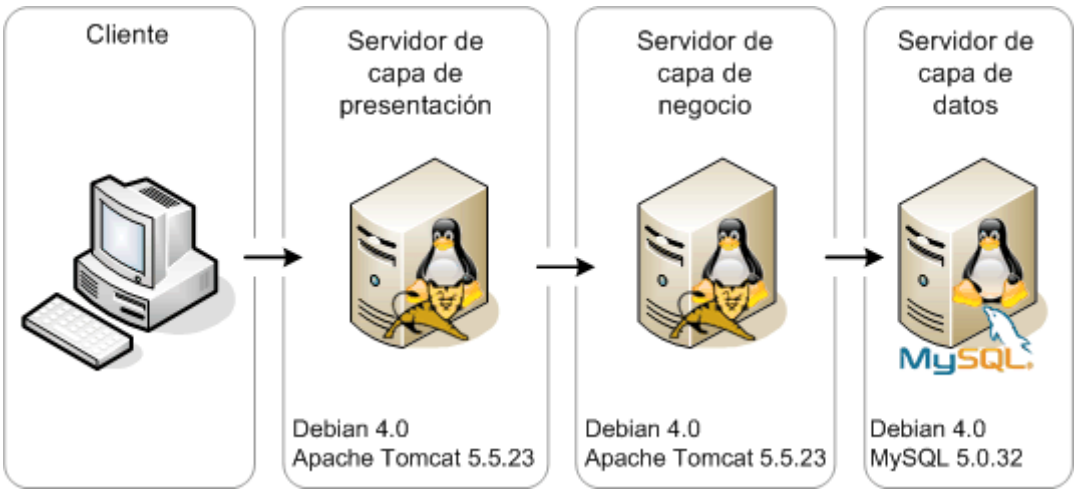

Figura 2.4. Arquitectura de despliegue de la aplicación Registro Centralizado de Donantes.

Además de la interfaz de usuario, el negocio de validación de la entrada de datos por formularios y otras funcionalidades para garantizar la usabilidad del sistema, la capa de presentación gestiona todo lo relacionado con la autenticación de los usuarios para lo que utiliza el componente de seguridad SAAA desarrollado para este tipo de aplicaciones. El proceso de autenticación genera información de estado que el usuario necesita para interactuar con la aplicación, la cual se almacena en la memoria RAM del servidor.

La capa de negocio contiene los servicios que dan cumplimiento a los requisitos funcionales del sistema, estos se implementan a través de servicios web que son utilizados por la capa de presentación y por otras aplicaciones.

La capa de datos contiene la base de datos relacional del sistema y el negocio necesario para garantizar la integridad referencial de los datos. A esta capa solo se accede desde la capa de lógica de negocio.

2.4. Descripción del despliegue actual de las aplicaciones

Los servidores sobre los que están desplegadas las aplicaciones mencionadas en el epígrafe 2.1 están ubicados físicamente en el nodo central de Infomed y son administrados, de manera remota, por los especialistas del Grupo de Servicios Remotos de la empresa Softel.

La configuración de la red de estos servidores está concebida para que utilicen direcciones IP reales. Además de todas las medidas de seguridad que implementan los administradores de la red de Infomed, cada servidor tienen implementado su propio cortafuego. El *software* que se utiliza para implementar del cortafuego de cada servidor es el Iptables y las políticas de seguridad que implementa el mismo solo permiten a los usuarios de la red del MINSAP acceder al servicio web que se presta en los servidores donde se ejecuta la capa de presentación de las aplicaciones.

La administración y monitoreo de los servidores se realiza desde varias estaciones de trabajo específicas, que se encuentran ubicadas físicamente en el local de los especialistas del Grupo de servicios remotos en la empresa Softel. El cortafuego individual de cada servidor está configurado para permitir labores de administración y monitoreo entre otras. Todas se realizan desde las estaciones de trabajo antes mencionadas.

El monitoreo, de las fallas en la red, los servidores y los servicios, se realiza utilizando el software Nagios en la versión 2.6. El software de monitoreo está configurado para que al detectar fallas notifique, utilizando varias vías, a los especialistas del grupo de servicios remotos y a otros que también están involucrados.

Además de esto, se está implementando el monitoreo del consumo de los recursos CPU, memoria RAM y tráfico de red, utilizando la herramienta MRTG (*Multi Router Traffic Grapher*) en la versión 2.14.7 combinada con el software RRDtool, versión 1.2.15. Este monitoreo solo permitirá conocer el comportamiento histórico del consumo de recursos en los servidores.

Acceso de los usuarios a las aplicaciones

Como resultado de una entrevista realizada a uno de los administradores de la red de Infomed se pudo conocer que el acceso, por parte de los usuarios de las aplicaciones, a la subred donde se ubican los servidores sobre los que se encuentran desplegadas estas, se realiza desde las subredes 201.220.192.0/19 y 10.0.0.0/8.

Aunque las políticas de seguridad que han configurado los administradores de la red de Infomed permiten que desde las subredes antes mencionadas se pueda acceder directamente a las aplicaciones, esto no ocurre así. La realidad es que los usuarios acceden a las aplicaciones a través de los servido-

res proxy que pueden estar ubicados en sus instituciones, en la provincia o el proxy nacional que se encuentra en la Ciudad de la Habana.

2.5. Conclusiones

Después de haber realizado el estudio de las aplicaciones médicas de tipo web que se desarrollan o se han desarrollado en la Facultad 7 de la UCI y/o Softel se arribó a las siguientes conclusiones:

1. Existen 25 aplicaciones médicas de tipo web que se encuentran en desarrollo o han sido desarrolladas por la Facultad 7 de la UCI y/o Softel que pueden necesitar esta solución.
2. Según el servidor de aplicación que utilizan y el número de niveles en que pueden ser desplegadas existen tres grupos de aplicaciones: los dos primeros grupos están compuestos por las utilizan el servidor de aplicación Apache + PHP y se despliegan en dos y tres niveles, las del tercer grupo se despliegan en tres niveles y utilizan el servidor de aplicación Apache-Tomcat.

CAPÍTULO 3. DISEÑO DE LA PROPUESTA DE SOLUCIÓN

Este capítulo está dedicado al diseño de una propuesta de solución genérica y un grupo de variantes específicas para las aplicaciones de muestra. En él también se realizan algunas sugerencias para el despliegue, gestión del monitoreo y gestión de la seguridad de la propuesta de solución.

3.1. Diseño del cluster de balanceo de carga genérico para propuesta de solución

Para la implementación del cluster de balanceo de carga para servidores web que se propone como solución, se utiliza el balanceador de carga IPVS que brinda el proyecto Linux Virtual Server. IPVS se complementa con el software Ldirectord para lograr una solución que implemente recuperación ante fallos en los servidores reales y el software Heartbeat alta disponibilidad. Esta se basa en la redundancia del recurso balanceador de carga.

El cluster está diseñado para que utilice el método de reenvío de paquetes LVS-DR. Se escogió este método porque logra que el balanceador de carga disminuya el consumo de sus recursos al permitir que los servidores web respondan directamente a los clientes y también aumenta la escalabilidad del sistema.

Pruebas previas no formalmente documentadas que se realizaron a uno de los diseños de la propuesta de solución para las aplicaciones médicas, permitieron comprobar que el método de reenvío de paquetes LVS-DR consume menos recursos que LVS-NAT. El recurso que más se consume cuando se utiliza este último es la red. La gráfica de la figura 3.1 muestra el consumo de recursos en el balanceador de carga cuando utiliza cada uno de estos métodos.

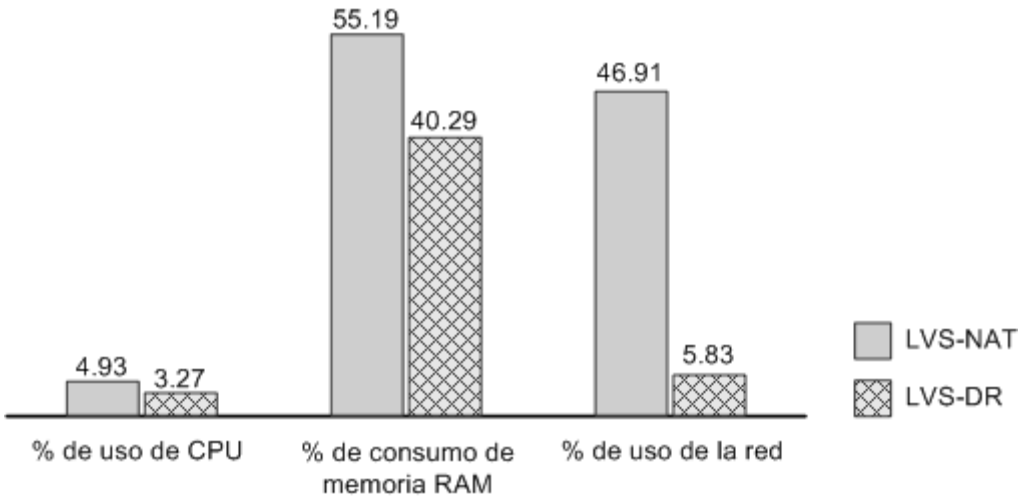

Figura 3.1. Consumo de recursos en el balanceador de carga cuando se utilizan los métodos de reenvío de paquetes LVS-NAT y LVS-DR.

Para la distribución de la carga entre los servidores reales se escogió el algoritmo menos conexiones con peso para, en caso que sea necesario, poder ponderar la carga entre los servidores web. Se seleccionó este algoritmo porque es muy utilizado en la implementación de este tipo de cluster, cuando cada conexión puede generar una carga de trabajo diferente.

El cluster genérico de la propuesta de solución está compuesto por un balanceador de carga primario, un balanceador de carga de respaldo y los servidores web necesarios para alcanzar la capacidad de procesamiento requerida para obtener el tiempo de respuesta deseado, como se muestra en la figura 3.2. Además está diseñado para que todos los servidores que lo componen utilicen la distribución Debian GNU/Linux 4.0 de este sistema operativo.

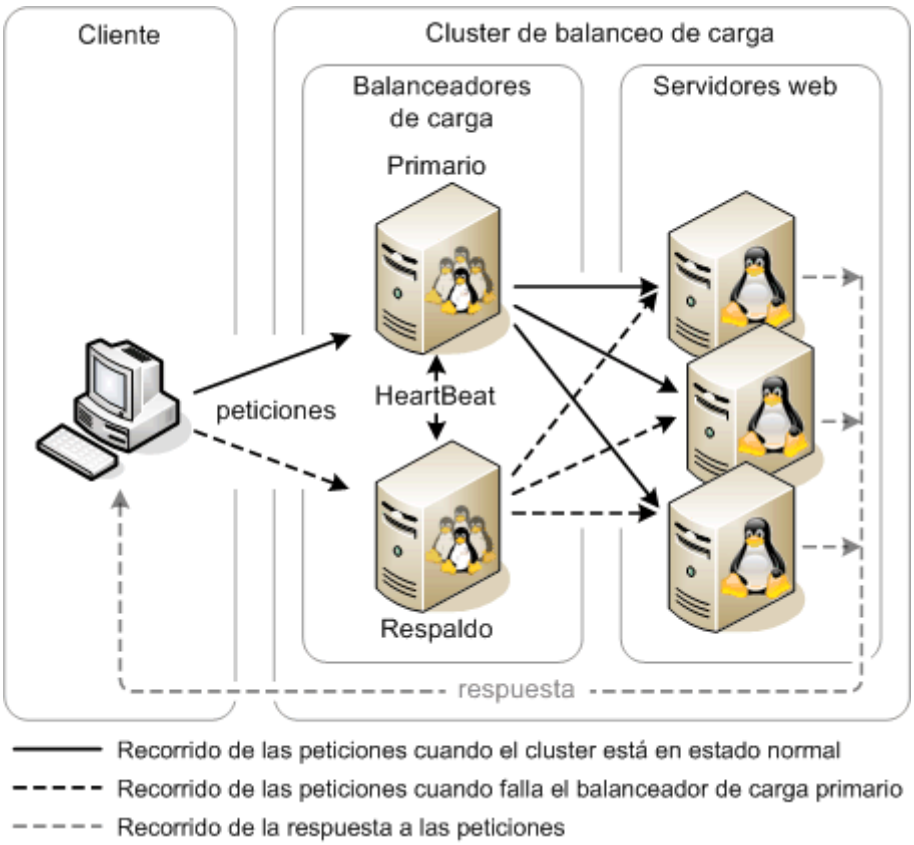

Figura 3.2. Estructura del cluster de balanceo de carga genérico. Las saetas indican el recorrido de las peticiones y el tráfico de respuesta.

Como requerimiento del método de reenvío de paquetes respuesta directa del servidor, la configuración de la red de todos los servidores del cluster debe permitir que sus direcciones IP pertenezcan al mismo dominio de *broadcast*. Aparte de la dirección IP de cada servidor, se necesita otra para que el cluster preste el servicio web. Esta última tiene que pertenecer al mismo dominio de *broadcast* que las de los servidores.

En los balanceadores de carga se necesita instalar los paquetes ipvsadm en la versión 1.24, ldirectord en la versión 1.2.5 y heartbeat en la versión 1.2.5.

En el fichero “ldirectord.cf”, que se encuentra ubicado en el directorio “/etc/ha.d/” de los balanceadores de carga, es donde se realiza la configuración del Ldirectord. En este fichero se define todo lo necesario para realizar el monitoreo del servicio que brindan los servidores reales y todos los parámetros requeridos para la configuración del balanceador de carga. La configuración del balanceador de carga la realiza dinámicamente el demonio del Ldirectord, a través de la herramienta Ipvsadm y para hacerlo utiliza la información que obtiene como resultado del chequeo a los servidores reales.

La configuración del Heartbeat, en ambos balanceadores de carga, se realiza en tres ficheros que se ubican en el directorio "/etc/ha.d/". El primero de estos ficheros se llama "authkeys" y en él se define: el método de autentificación y la clave que utilizan los demonios del HearBeat, que corren en los balanceadores, para autenticarse entre ellos. Este fichero debe ser el mismo en ambos nodos. El segundo es "ha.cf", en el se especifica la dirección IP que debe chequear el demonio del Heartbeat de cada balanceador, la forma de comunicación y todos los parámetros de configuración se utilizarán. El tercer fichero se llama "haresources" y en el es donde se especifica el balanceador de carga preferido para prestar servicio y los recursos sobre los que tendrá control el demonio del Heartbeat.

Los recursos sobre los que toma el control el demonio del Heartbeat son la VIP, el *script* llamado "hadr-master" que se encuentra ubicado en el directorio "/etc/init.d/" y se utiliza para modificar algunas variables del kernel necesarias para el correcto funcionamiento del balanceador de carga y el demonio del Ldirectod.

En los servidores web no es necesario instalar ningún software además de los que necesitan para prestar el servicio. Como en el diseño del cluster se está utilizando respuesta directa del servidor como método de reenvío de paquetes, los servidores reales, además de su dirección IP, necesitan tener configurada alguna interfaz de red con la VIP. Esta interfaz no debe responder a las peticiones de ARP del enrutador u otro servidor que se encuentre en la misma subred.

La configuración de la VIP en los servidores reales, para lo que se utiliza un alias de la interfaz loopback y la modificación de las variables del kernel para evitar que esta interfaz responda a las peticiones ARP, se realiza mediante un *script* llamado "dr-server" que se ubica en el directorio "/etc/init.d/" y se ejecuta automáticamente cuando se levanta el sistema.

Incluir un sistema de archivos distribuido en el cluster de balanceo de carga

Para implementar el sistema de archivos distribuido que necesitan algunas variantes de la propuesta de solución para almacenar los ficheros que deben estar accesibles y actualizados para todos los servidores web que componen el cluster se utiliza el software NFS.

El nfs-server-kernel en la versión 1.0.10 es el software que se necesita para implementar el servidor del sistema de archivos distribuido. Para no utilizar más servidores y evitar que este constituya un único punto de fallo para el cluster, se monta junto a ambos balanceadores de carga. El servicio se presta por una dirección IP diferente a la VIP, que pertenece al mismo dominio de *broadcast* que esta y sobre la cual solo tiene control el demonio del Heartbeat.

La configuración del servidor NFS se realiza a través del fichero “exports” que se encuentra ubicado en el directorio “/etc”. En este fichero se define el directorio que se desea exportar, quienes tienen acceso a él y los permisos con los que accederá cada cual.

Para que los servidores web puedan acceder y montar automáticamente los directorios que exporta el servidor NFS, necesitan instalar el nfs-common en la versión 1.0.10 y el autofs en la versión 4.1.4. Toda la configuración se realiza en dos ficheros que se encuentra en “/etc/”. El primero de estos ficheros se llama “auto.mater” y el otro se define en él.

Normalmente los servidores web acceden al sistema de archivos distribuido que está ubicado junto al balanceador primario, figura 3.3 a). Cuando falla el balanceador de carga primario y el de respaldo toma su lugar, el servidor del sistema de archivos distribuido también es migrado, pero los servidores reales no pueden utilizarlo aunque se sigue prestando por la misma dirección IP.

Cuando el balanceador de carga de respaldo es el que presta servicio, al sistema de archivos distribuido que se accede, es el que está montado junto a el, figura 3.3 b). Si el balanceador de carga primario toma su lugar, todos los servicios vuelven a ser migrados hacia el y los servidores web no pueden seguir accediendo al sistema de archivos distribuido.

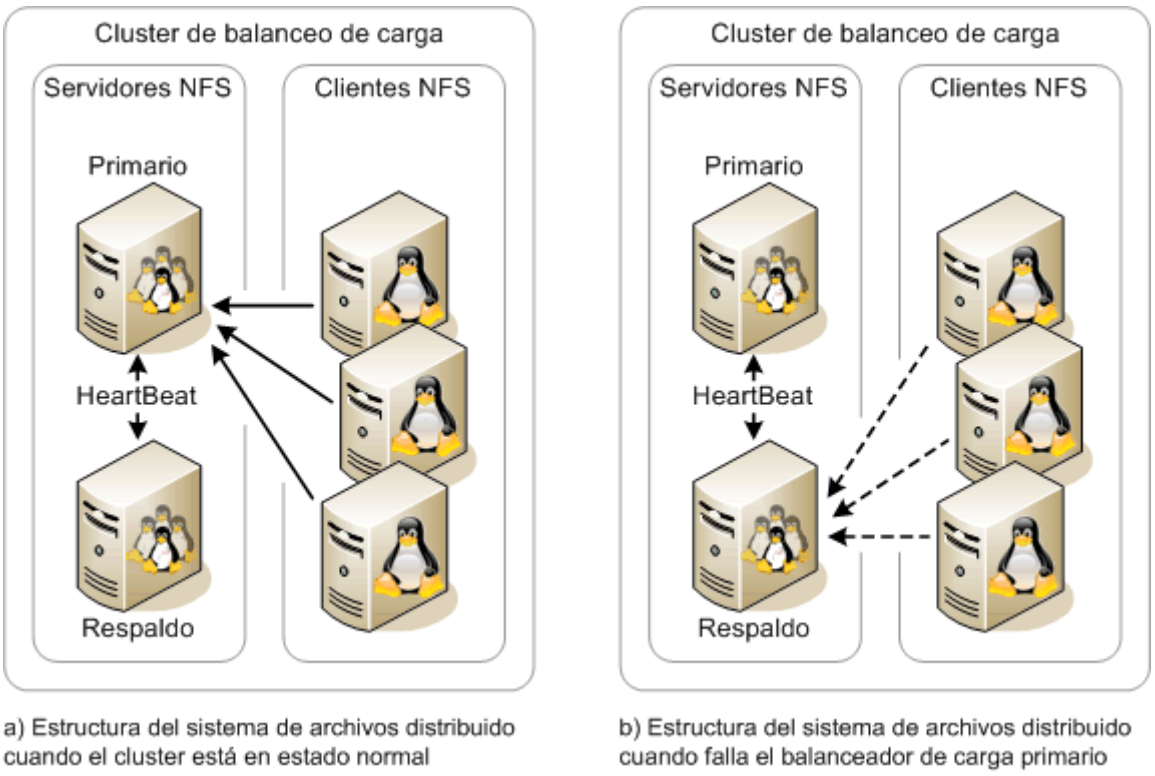

Figura 3.3. Estructura del sistema de archivos distribuido.

Para solucionar este problema se decidió crear un *script*, sobre el cual solo tenga control el demonio del Heartbeat del balanceador de carga de respaldo, para que cada vez que los recursos sean migrados de un balanceador de carga a otro mande a los servidores web a reiniciarse. De esta forma cuando los servidores web se levantan no saben donde estaba el sistema de archivos distribuido antes y pueden acceder a él en su nueva ubicación. Este mecanismo tiene como principal inconveniente, hacer que el proceso de recuperación sea más lento.

La descripción de la instalación y configuración de este diseño genérico de la propuesta de solución se describe en el documento que se adjunta en formato digital titulado “Instalación y configuración de un cluster de balanceo de carga genérico para servidores web”.

3.2. Diseño de las variantes de la propuesta de solución para la aplicación RPS

Dado que la aplicación Registro de personal de la salud puede ser desplegada en tres niveles independientes, de los cuales dos son sobre son sobre un servidor web, existen tres variantes de la propuesta de solución para esta aplicación.

La primera variante consiste en sustituir el servidor de capa de presentación por un cluster de balanceo de carga, la segunda en sustituir el servidor de lógica de negocio igual que en la primera variante y la tercera en integrar las dos primeras variantes para que cada una de estás capas se ejecute sobre un cluster de balanceo de carga.

Primera variante de la propuesta de solución para la aplicación RPS

La primera variante de la propuesta de solución para la aplicación RPS se diseña para el caso que el servidor de capa de presentación no sea capaz de soportar la carga generada por los niveles de concurrencia que debe gestionar y consiste en sustituir dicho servidor por el cluster de balanceo de carga para servidores web que se describe en el epígrafe 3.1.

Como el servidor de capa de presentación de la aplicación RPS almacena en su disco duro todos los ficheros de sesión que genera el PHP, es necesario definir la forma en que el cluster va a gestionar esta información para no afectar el funcionamiento correcto de la aplicación.

Los ficheros de sesión que genera el PHP tienen un tiempo de vida útil relativamente corto y pueden ser utilizados solo por el servidor de capa de presentación donde se crean. Utilizar la persistencia de sesión basada en dirección IP origen que permite el balanceador de carga podría garantizar el correcto funcionamiento de la aplicación. Debido a que, para el balanceador de carga, el acceso de los usuarios a la aplicación se realiza desde la dirección IP de los servidores proxy que estos utilizan para navegar, esta alternativa puede afectar el equilibrio en la distribución de carga.

Para que no se afecte el balanceo de carga se decidió que el cluster incluya un sistema de archivos distribuido para almacenar los ficheros de sesiones que genera el PHP. En la figura 3.4 se muestra el diseño final del cluster de balanceo de carga para esta variante de la propuesta de solución.

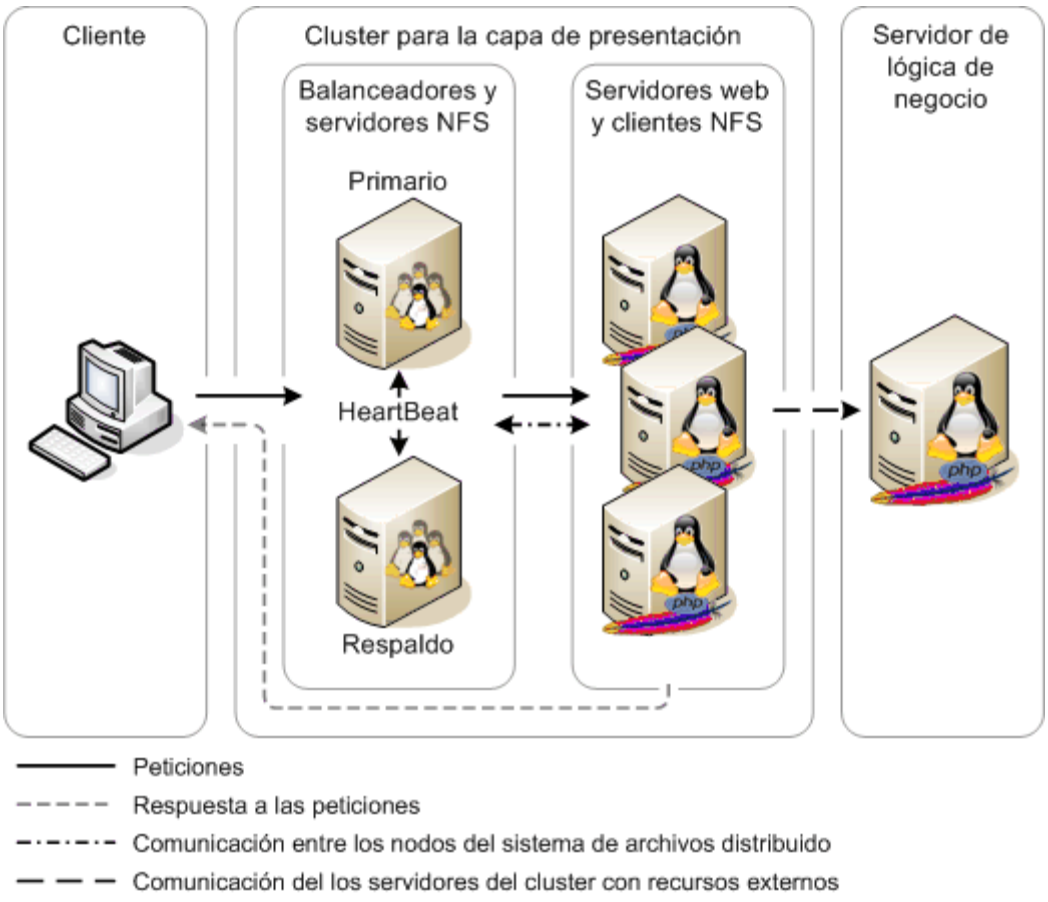

Figura 3.4. Diseño final del cluster de balanceo de carga para la capa de presentación de la aplicación RPS.

El proceso de instalación y configuración del cluster para esta variante de la propuesta de solución se describe en el documento “Cluster de balanceo de carga para la capa de presentación de la aplicación Registro de personal de la salud”, el cual se adjunta en formato digital.

La instalación y configuración inicial de los servidores de capa de presentación se realiza de la misma manera en todos y como se indica en el documento de despliegue correspondiente a la aplicación Registro de personal de la salud.

Para que la capa de presentación se ejecute correctamente sobre el cluster, la configuración es necesario realizarla teniendo en cuanta los siguientes aspectos:

- La configuración de los *virtualhosts* del Apache deben permitir el acceso a la aplicación solo por la VIP y el chequeo que realiza el balanceador de carga por la dirección IP correspondiente a cada servidor.
- Los ficheros de sesión que genera el PHP deben almacenarse en el sistema de archivos distribuido.

Segunda variante de la propuesta de solución para RPS

La segunda variante de la propuesta de solución para la aplicación RPS es muy semejante a la primera variante y se utiliza cuando el servidor de lógica de negocio no es capaz de soportar la carga generada por las peticiones que realiza la capa de presentación u otras aplicaciones que consumen los servicios web que en ella se implementan. Esta variante se basa en sustituir el servidor por un cluster de balanceo e carga para servidores web con las características del que se describe en el epígrafe 3.1.

En el servidor sobre el que se ejecuta la capa de lógica de negocio de la aplicación se almacenan temporalmente en disco los reportes que se generan en formato PDF y EXCEL. La utilización de esta funcionalidad se realiza desde la capa de presentación.

Para garantizar que en el momento de descargar el reporte, la capa de presentación haga la petición al mismo servidor que lo generó, podría utilizarse persistencia de sesión basada en direcciones IP origen. Si la capa de presentación está desplegada sobre un solo servidor el cluster de balanceo de carga deja de cumplir su función, pues todas las peticiones que realiza siempre serán atendidas por un mismo servidor, si está desplegada también sobre un cluster de balanceo de carga puede afectar el equilibrio del balanceo.

Para no afectar el balanceo de carga del cluster para la capa de lógica de negocio y lograr que los reportes en formato PDF y EXCEL estén disponibles para todos los nodos del cluster se implementa un sistema de archivos distribuido para almacenarlos.

El diseño final del cluster de balanceo de carga para la capa de lógica de negocio de la aplicación RPS se muestra en la figura 3.5.

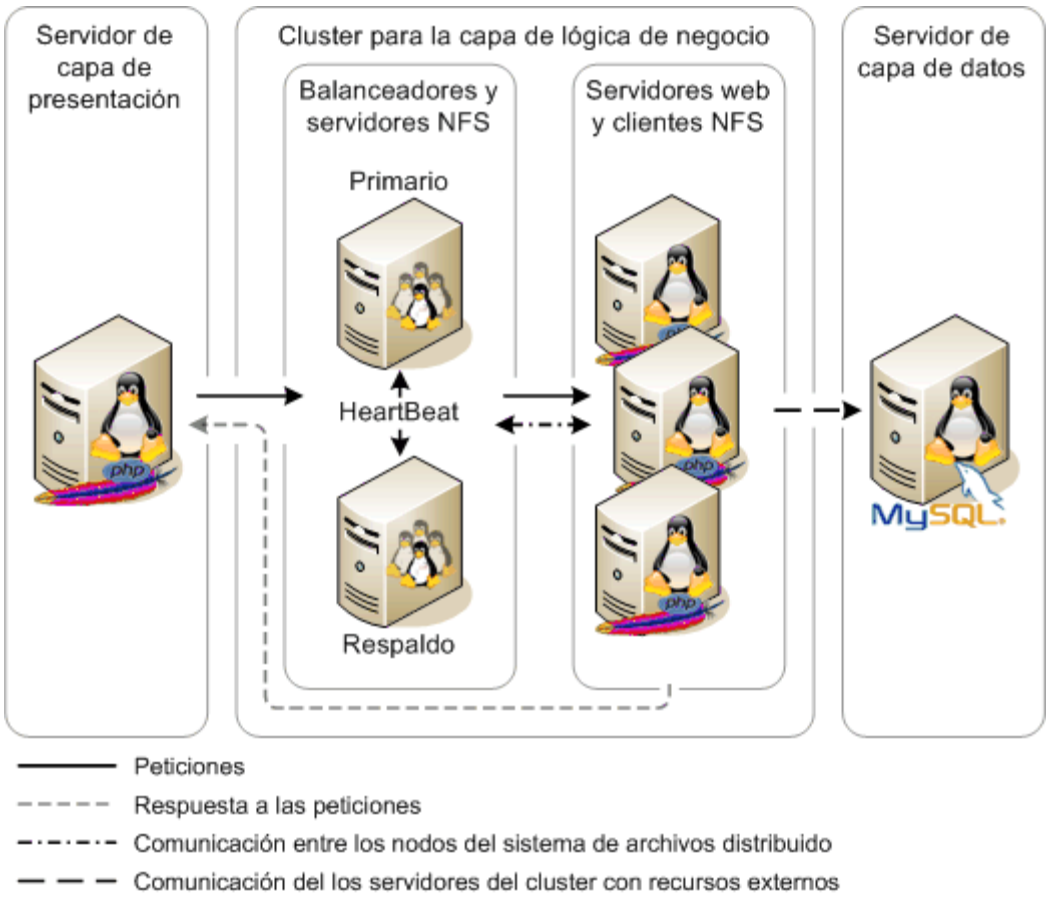

Figura 3.5. Diseño final del cluster de balanceo de carga para la capa de lógica de negocio de la aplicación RPS.

La descripción detallada de la instalación y configuración del cluster para esta variante de la propuesta de solución se realiza en el documento “Cluster de balanceo de carga para la capa de lógica de negocio de la aplicación Registro de personal de la salud” el cual se adjunta en formato digital.

La instalación y configuración de los servidores de lógica de negocio se realiza de la misma manera en todos y como se indica en el documento de despliegue correspondiente a la aplicación Registro de personal de la salud.

Para que esta capa se ejecute correctamente sobre el cluster, la configuración es necesario realizarla teniendo en cuanta los siguientes aspectos:

- La configuración de los *virtualhosts* del Apache deben permitir el acceso a la lógica de negocio solo por la VIP y el chequeo que realiza el balanceador de carga por la dirección IP correspondiente a cada servidor.
- Los reportes en formato PDF y EXCEL deben almacenarse en el sistema de archivos distribuido.
- Al cambiarse el directorio donde se almacenan los reportes, el *virtualhost* que se utiliza para acceder a ellos debe ser actualizado.

Tercera variante de la propuesta de solución para RPS

La tercera variante de la propuesta de solución para la aplicación RPS consiste en ejecutar la capa de presentación y la capa de lógica de negocio sobre cluster de balanceo de carga cada una. En la imagen de la figura 3.6 se trata de mostrar como quedaría desplegada la aplicación.

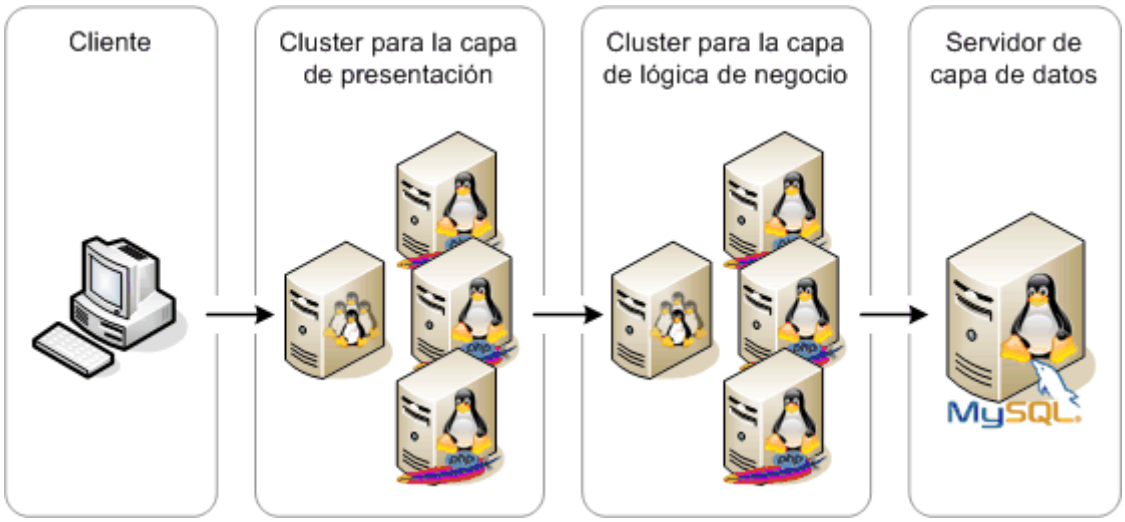

Figura 3.6. Despliegue de la aplicación RPS sobre cluster en las capas de presentación y lógica de negocio.

El cluster que se utiliza para cada una de las capas es el mismo que se diseño como una solución independiente en las variantes anteriores de la propuesta de solución para esta aplicación. Al cada cluster funcionar como un servidor virtual, esta variante de la propuesta de solución no es más que la integración, en un mismo despliegue, de las antes descritas.

3.3. Diseño de la variante de la propuesta de solución para la aplicación Balance Material

El despliegue de la aplicación Balance de Material se puede realizar en dos niveles, uno para las capas de presentación y lógica de negocio y otro para la capa de datos. Como solo se utiliza un servidor web en el despliegue de esta aplicación, existe una sola variante de la propuesta de solución para ella.

La variante de la propuesta de solución para esta aplicación se diseña para cuando el servidor web, sobre el que se ejecutan las capas de presentación y lógica de negocio, no es capaz de soportar la carga que generan los usuarios al navegar por la aplicación. Esta variante consiste en remplazar el servidor por un cluster de balanceo de carga para servidores web con las mismas características del que se describió en el epígrafe 3.1.

En el sistema de ficheros del servidor aplicación se almacenan los ficheros de sesión que genera el PHP. Esta situación crea el mismo problema que se analizó en el diseño de la primera variante de la propuesta de solución para la aplicación Registro de personal de la salud y dado que las circunstancias son las mismas, la solución también lo será.

Por tanto, para que no se afecte el balanceo de carga se decidió que el cluster de balanceo de carga para la aplicación Balance de Material también incluya un sistema de archivos distribuido para almacenar los ficheros de sesiones que genera el PHP.

En la figura 3.7 se muestra el diseño final del cluster de balanceo de carga para la aplicación Balance de Material.

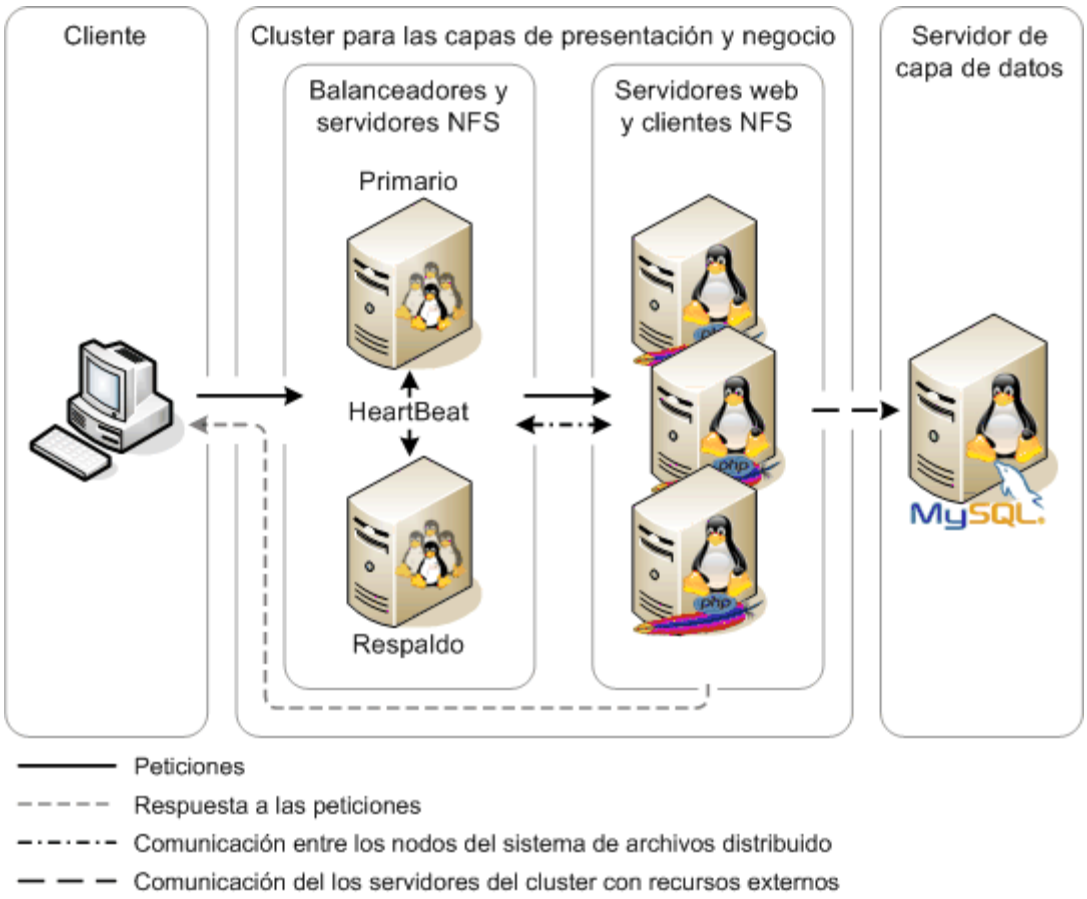

Figura 3.7. Diseño final del cluster de balanceo de carga para la aplicación Balance Material.

En el documento que se adjunta en formato digital y se titula “Cluster de balanceo de carga para la aplicación Balance Material” se describe en detalle el proceso de instalación y configuración del cluster para esta variante de la propuesta de solución.

La instalación y configuración de los servidores de aplicación se realiza de la misma manera en todos y como se indica en el documento de despliegue correspondiente a la aplicación Balance Material.

Para que las capas de aplicación y negocio se ejecuten correctamente sobre el cluster, la configuración es necesario realizarla teniendo en cuanta los siguientes aspectos:

- La configuración de los *virtualhosts* del Apache deben permitir el acceso a la aplicación solo por la VIP y el chequeo que realiza el balanceador de carga por la dirección IP correspondiente a cada servidor.

- Los ficheros de sesión que genera el PHP deben almacenarse en el sistema de archivos distribuido.

3.4. Diseño de las variantes de la propuesta de solución para la aplicación RCD

Dado que, igual a la aplicación RPS, la aplicación Registro centralizado de donantes se despliega en tres niveles independientes, de los cuales dos son sobre un servidor web, existen tres variantes de la propuesta de solución para la misma.

La primera y segunda variante consiste en sustituir los servidores de capa de presentación y lógica de negocio por un cluster de balanceo de carga cada uno y la tercera en integrar las dos variantes anteriores.

Primera variante de la propuesta de solución para la aplicación RCD

La primera variante de la propuesta de solución para la aplicación Registro centralizado de donantes tiene las mismas características que la variante de la propuesta de solución diseñada para la capa de presentación de la aplicación RPS y es utilizada cuando el servidor de capa de presentación no es capaz de soportar la carga que generan los usuarios al navegar sobre la aplicación. La estructura del cluster de balanceo de carga para esta aplicación es la que describió en el epígrafe 3.1.

Por defecto el servidor de aplicaciones Apache-TomCat sobre el que se ejecuta la capa de presentación de la aplicación RCD guarda la información de estado de las sesiones de usuarios en memoria RAM. Debido a esto, para que la aplicación funcione correctamente, es necesario que cada usuario sea atendido siempre por el mismo servidor, lo que se puede lograr utilizando persistencia de sesión basada en dirección IP origen. Como esta aplicación, al igual que RPS y Balance Material, es accedida desde un reducido número de direcciones IP, el diseño de esta solución no garantiza un balanceo de carga efectivo.

En la figura 3.8 se muestra el diseño final del cluster de balanceo de carga para la capa de presentación de la aplicación RCD.

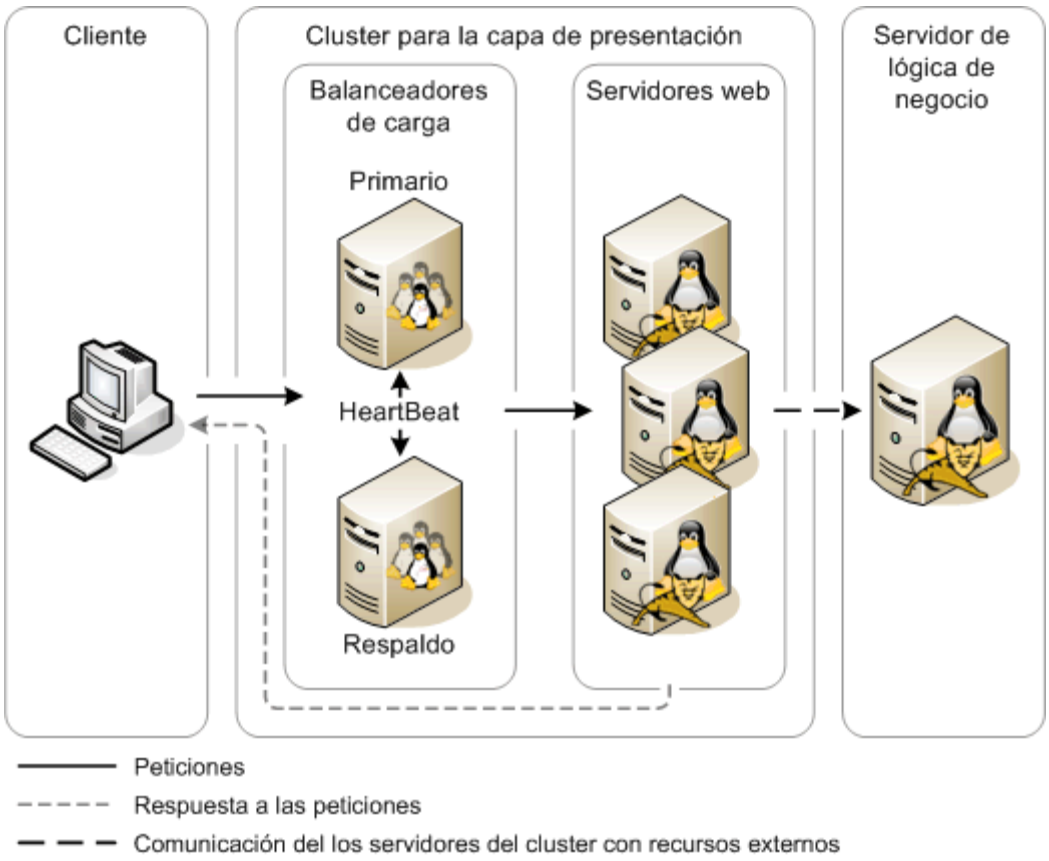

Figura 3.8. Diseño final del cluster de balanceo de carga para la capa de presentación de la aplicación RCD.

El proceso de instalación y configuración del cluster para esta variante de la propuesta de solución está descrito en el documento “Cluster de balanceo de carga para la capa de presentación de la aplicación Registro centralizado de donantes” el cual se adjunta en formato digital.

La instalación y configuración de los servidores de capa de presentación se realiza de la misma manera en todos y como se indica en el documento de despliegue correspondiente a la aplicación Registro centralizado de donantes.

Para que la capa de presentación se ejecute correctamente sobre el cluster, la configuración es necesario realizarla teniendo en cuenta que la configuración del Apache-TomCat debe permitir el acceso a la aplicación solo por la VIP y el chequeo que realiza el balanceador de carga por la dirección IP correspondiente a cada servidor.

Segunda variante de la propuesta de solución para la aplicación RCD

La segunda variante de la propuesta de solución para la aplicación Registro centralizado de donantes se realiza para en caso de que el servidor de lógica de negocio no sea capaz de soportar la carga generada por las peticiones que realiza la capa de presentación, esta variante se basa en sustituir el servidor en cuestión por un cluster de balanceo e carga para servidores web como el que se describe en el epígrafe 3.1.

En esta variante se modifica la arquitectura de despliegue para que la capa de lógica de negocio se ejecute sobre un cluster de balanceo de carga y las otras dos capas se mantengan ejecutándose sobre servidores independientes. En la figura 3.9 se muestra el diseño final del cluster de balanceo de carga para esta variante de la propuesta de solución.

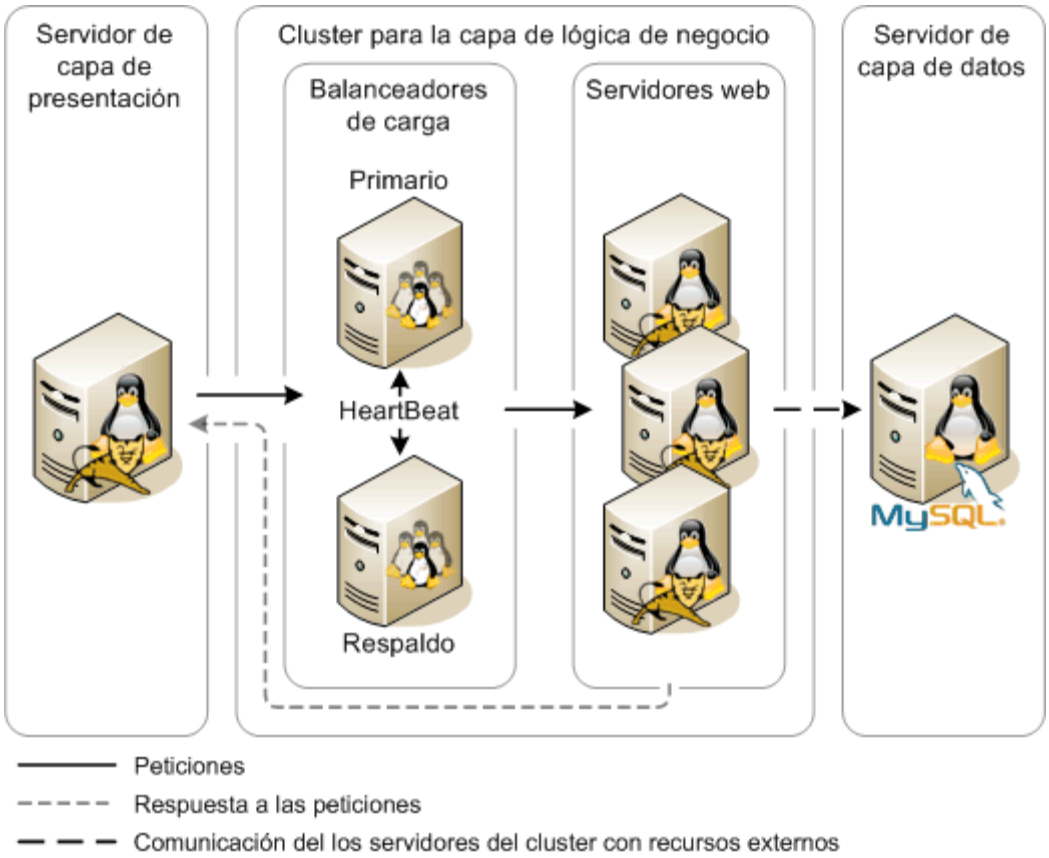

Figura 3.9. Diseño final del cluster de balanceo de carga para la capa de lógica de negocio de la aplicación RCD.

Todo el proceso de instalación y configuración del cluster se describe en el documento “Cluster de balanceo de carga para la capa de lógica de negocio de la aplicación Registro centralizado de donantes” el cual se adjunta en formato digital.

La instalación y configuración de los servidores de lógica de negocio se realiza de la misma manera en todos y como se indica en el documento de despliegue correspondiente a la aplicación Registro centralizado de donantes.

Para que la capa de lógica de negocio se ejecute correctamente sobre el cluster, la configuración es necesario realizarla teniendo en cuanta que la configuración del Apache-TomCat debe permitir el acceso a esta capa solo por la VIP y el chequeo que realiza el balanceador de carga por la dirección IP correspondiente a cada servidor.

Tercera variante de la propuesta de solución para la aplicación RCD

La tercera variante de la propuesta de solución para la aplicación RCD consiste en ejecutar la capa de presentación y la capa de lógica de negocio sobre cluster de balanceo de carga cada una. En la imagen de la figura 3.10 se esboza cómo quedaría desplegada la aplicación.

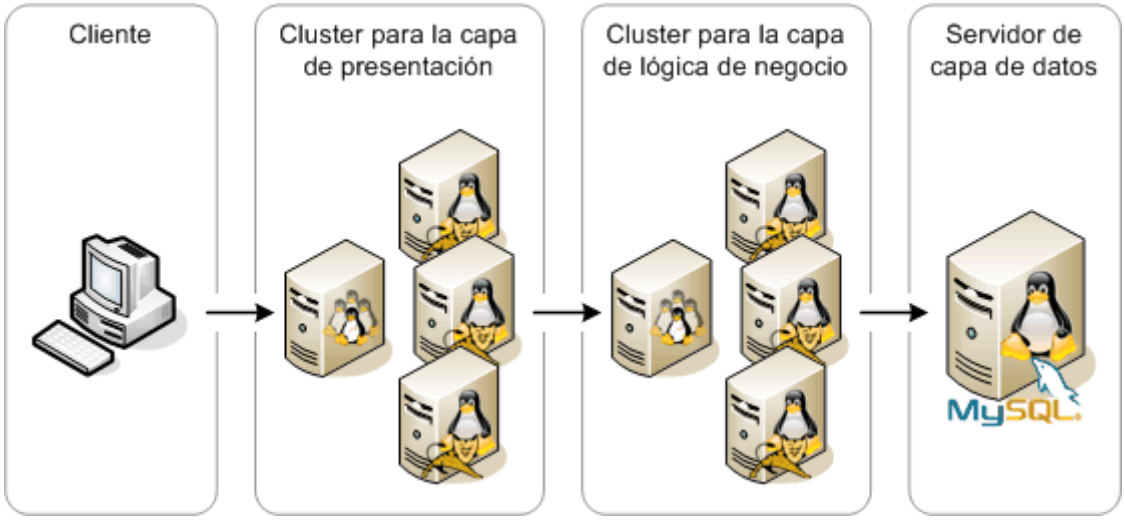

Figura 3.10. Despliegue de la aplicación RCD sobre cluster en las capas de presentación y lógica de negocio.

El cluster que se utiliza para cada una de las capas es el mismo que se diseño como una solución independiente en las variantes anteriores de la propuesta de solución para esta aplicación. Al cada

cluster funcionar como un servidor virtual, esta variante de la propuesta de solución no es más que la integración, en un mismo despliegue, de las antes descritas.

3.5. Consideraciones para el despliegue de las aplicaciones

Implementar alguna o todas las variantes de la propuesta de solución para las aplicaciones médicas centralizadas introduce modificaciones en el despliegue y gestión de los sistemas. Algunos de los cambios más significativos que es necesario hacer es el reajuste en el diseño de la red, en los métodos de gestión del monitoreo y en las políticas que se utilizan para la gestión de la seguridad.

Propuesta de reajuste el diseño de la red

Los servidores sobre los que están desplegadas algunas aplicaciones desarrolladas por la facultad 7 de la UCI y/o la empresa productora de software Softel en Infomed, necesitan estar accesibles para los usuarios que utilizan las aplicaciones y para los administradores.

Como el acceso a las aplicaciones y la gestión de los sistemas se realizan desde subredes externas a la subred de *housing* de Infomed y el número actual de servidores es pequeño, estos están configurados con direcciones IP reales para facilitar estos procesos.

Debido a la arquitectura del tipo de cluster de balanceo de carga que se utiliza en las propuestas de soluciones, todos los servidores que lo componen tienen que estar configurados para pertenecer al mismo dominio de *broadcast*.

Si en los despliegues de las aplicaciones se comienza a sustituir cada servidor web por un cluster de balanceo de carga, por la necesidad de mejorar los tiempos de respuesta del sistema, el número de servidores involucrados aumenta y con ello el número de direcciones IP reales que serían necesarias. Este proceso sería insostenible debido a la escasez de estás últimas.

Para solucionar este problema se propone reajustar el diseño de la red y ubicar a todos los servidores en una subred IP privada. Para que los usuarios de las aplicaciones web puedan utilizarlas, en el enrutador que constituye el punto de acceso a la subred privada se debe realizar una traducción de las direcciones de red para asociar la dirección IP real por el que se accede a cada aplicación, con la

dirección IP privada por la que presta servicio la capa de presentación de la misma. Esta configuración en el enrutador también puede ser necesaria para permitir el acceso a ciertos servicios que se prestan en la capa de lógica de negocio.

También se propone implementar un *tunneling* VPN (*Virtual Private Network*) entre las puertas de enlace de las redes donde se encuentran los servidores y las estaciones de monitoreo y administración, para permitir estos procesos.

Propuestas de modificaciones a la gestión del monitoreo

Actualmente se utiliza la herramienta Nagios para el monitoreo de fallas en la red, los servidores y los servicios y la herramienta MRTG para conocer el consumo de recursos en cada servidor de manera histórica.

Cuando se comiencen a implementar las variantes de las propuesta de solución, todos los servidores que componen cada cluster serán incluidos en este monitoreo. Pero además se propone monitorear el rendimiento de cada cluster e implementar otra forma del monitoreo del consumo de recursos.

Monitoreo del rendimiento de un cluster

Para implementar el monitoreo del rendimiento de un cluster de balanceo de carga para servidores web se propone crear una página que implique cierto procesamiento y se monitoree el comportamiento del tiempo de respuesta del sistema para dicha página.

El objetivo del monitoreo es que el *software* que se utiliza para este propósito notifique a los administradores cuando el rendimiento del cluster sea preocupante y/o crítico. Para lograr esto es necesario definir dos niveles de umbral para el tiempo de respuesta del sistema para la página de monitoreo. El primer nivel de umbral sería para indicar que el rendimiento del cluster ha comenzado a deteriorarse y alcanza niveles preocupantes por lo que se deben tomar medidas preventivas. Cuando se alcanza el sungo nivel de umbral es porque se ha llegado a un estado crítico en el rendimiento del cluster.

Los niveles de umbral en el tiempo de respuesta del sistema se sugiere que sean definidos mediante un estudio del comportamiento histórico de esta métrica para la página de monitoreo. Además se re-

comienda que el estudio se realice sobre una configuración específica del sistema en un ambiente de producción real.

Para incorporar este monitoreo al que ya se realiza con el *software* Nagios se propone programar un *script* que realice un número determinado de peticiones, calcule el valor promedio del tiempo de respuesta y basándose en este evalúe el estado del rendimiento en el cluster.

Monitoreo del consumo de recursos en los servidores

El monitoreo para detectar niveles críticos en el consumo de recursos en alguno de los servidores es muy semejante a la del monitoreo del rendimiento del sistema.

Para este, también se definen dos niveles de umbrales específicos para el consumo de los recursos CPU y memoria RAM en cada servidor. La respuesta del software de monitoreo al detectar que el consumo de recursos has alcanzado alguno de los niveles de umbral sigue siendo notificar a los administradores del sistema.

Para definir cuando se alcanzan niveles críticos en el consumo de recursos en los servidores es necesario conocer el comportamiento histórico de las métricas % de uso de CPU y % de consumo de memoria RAM en cada uno de ellos. Se propone que para esto se realice un estudio de esas métricas en el entorno de producción.

Igual que en el monitoreo del rendimiento del cluster se propone programar un *script* que realice esta labor. Para incorporar este monitoreo al que ya se realiza, el *script* debe haberse programado siguiendo las normas que define el *software* Nagios.

Propuestas de modificaciones a la gestión de la seguridad

Las políticas de seguridad, relativas a la comunicación entre los servidores, que implementan los cortafuegos individuales de cada servidor solo permiten conexiones al servicio que presta, a los servidores que lo necesiten.

Estas implementaciones de seguridad también serán realizadas a los servidores que componen cada cluster de balanceo de carga que se implemente. Para implementar esta política de seguridad sin que se afecte la comunicación entre los servidores de cada cluster es necesario conocer que:

- Para el chequeo de estado, los balanceadores de carga se comunican entre ellos utilizando el protocolo UDP.
- El balanceador de carga activo le reenvía a los servidores web los paquetes pertenecientes a las conexiones TCP, que los clientes tienen establecidas con ellos, para poder acceder al servicio HTTP.
- El balanceador de carga activo establece conexiones TCP con los servidores web para el monitoreo del servicio HTTP.
- Cuando se utiliza sistema de archivos distribuido, los clientes NFS, que se encuentran junto a los servidores web, establecen conexiones TCP con el servidor NFS, que está montado junto al balanceador de carga activo.

3.6. Mejoras al diseño del cluster para el Entorno Virtual de Aprendizaje de la UCI

En el mes de junio del año 2007 el autor de esta investigación realizó el diseño de un cluster de balanceo de carga para servidores web Apache con el objetivo de ejecutar sobre él la aplicación que soporta el Entorno Virtual de Aprendizaje (EVA) de la UCI. El cluster fue desplegado en el mes de julio de ese mismo año.

El cluster para EVA UCI se diseñó principalmente para soportar el nivel de concurrencia, de aproximadamente 2000 usuarios, que se genera en los momentos en que realizan pruebas en la aplicación los estudiantes de todo un año de la universidad. Un objetivo secundario del diseño era aumentar el rendimiento de la aplicación en el uso cotidiano (Peña Montero et al. 2007).

Aunque el cluster para servidores web tiene la capacidad para soportar poco más de 2000 usuarios navegando concurrentemente y actualmente está en explotación brindando buen servicio, el principal objetivo no se pudo logar. Esto se debe a que la capacidad de procesamiento del servidor de base de datos que utiliza la aplicación no es suficiente para soportar la carga que ella genera al ejecutarse sobre el cluster de balanceo de carga.

Descripción del cluster de balanceo de carga para EVA UCI

El diseño del cluster de balanceo de carga de servidores web para EVA UCI se realizó para utilizar el balanceador de carga IPVS del proyecto Linux Virtual Server combinado con el software del Ldirectord del proyecto Linux HA para garantizar la recuperación ante fallos en los servidores reales. Este utiliza respuesta directa como método de reenvío de paquetes a los servidores reales y menos conexiones con peso como algoritmo de distribución de carga. Además el cluster incluye un sistema de archivos distribuido para almacenar los ficheros de sesión que genera el PHP y el directorio de datos de la aplicación.

El cluster se compone de 10 servidores, que se dividen en el balanceador de carga y nueve servidores web. También existe un servidor de datos, donde está ubicado el sistema de archivos distribuido que utiliza el cluster y la base de datos de la aplicación. Todos los servidores que componen el cluster tienen instalada la distribución Debian GNU/Linux 4.0 de este sistema operativo (Peña Montero et al. 2007).

El balanceador de carga tiene instalado el ipvsadm (1.24) y el ldirectord (1.2.5). El primero para la administración del balanceador de carga IPVS y el segundo para el monitoreo de los servidores reales y la configuración dinámica del balanceador de carga, lo que permite la recuperación ante fallos en los servidores reales.

En el servidor de datos está instalado el nfs-kernel-server (1.0.10) para implementar el sistema de archivos distribuido y el mysql-server-5.0 (5.0.32) para la base de datos que utiliza la aplicación.

Los servidores reales, además de todo lo que necesitan para soportar la aplicación, tienen instalado el nfs-common (1.0.10) para acceder al sistema de archivos distribuido que está montado en el servidor de datos y el autofs (4.1.4) para montar automáticamente y a demanda los recursos que en este se exportan.

En uno de los servidores reales se encuentra instalado y configurado un balanceador de carga de respaldo. Este se encuantra inactivo, para en caso que el balanceador de carga primario quede fuera de servicio tomar su lugar. El proceso de intercambiar los balanceadores de carga es necesario que lo realice manualmente el administrador del sistema.

Mejoras al diseño del cluster de balanceo de carga para EVA UCI

En el diseño del cluster de balanceo de carga para EVA UCI tiene dos debilidades: el balanceador de carga y el servidor de datos, pues ambos constituyen puntos críticos de fallos que pueden provocar la caída del sistema completo.

El estudio realizado para el diseño de las variantes de la propuesta solución, permite implementar alta disponibilidad en el cluster para EVA UCI y eliminar el punto crítico de fallo que constituye el balanceador de carga.

Para la implementación de alta disponibilidad en el balanceador de carga se va a utilizar el software Heartbeat, del proyecto Linux HA, el cual permite obtener de alta disponibilidad IP de tipo activo/pasivo.

La alta disponibilidad se basa en la redundancia de recursos, por lo que para implementar alta disponibilidad en el balanceador de carga es necesario dos balanceadores. En esta solución se va a utilizar como balanceador de carga de respaldo el que se encuentra inactivo en uno de los servidores reales.

En ambos balanceadores es necesario instalar y configurar el Hearbeat para que su demonio tome el control de los recursos que es necesario migrar y realice un chequeo constante del estado del otro balanceador de carga.

Cuando el cluster se encuentra en estado normal, como se muestra en la figura 3.11, el balanceador de carga de respaldo desempeña dos roles, pues chequea el estado del balanceador de carga primario y presta servicios como un servidor web más. Por este motivo es necesario que la carga de trabajo del servidor web en cuestión sea menor que la del resto de los servidores. Para lograrlo es necesario especificarle al balanceador de carga primario que el peso de este servidor es menor que el de los otros.

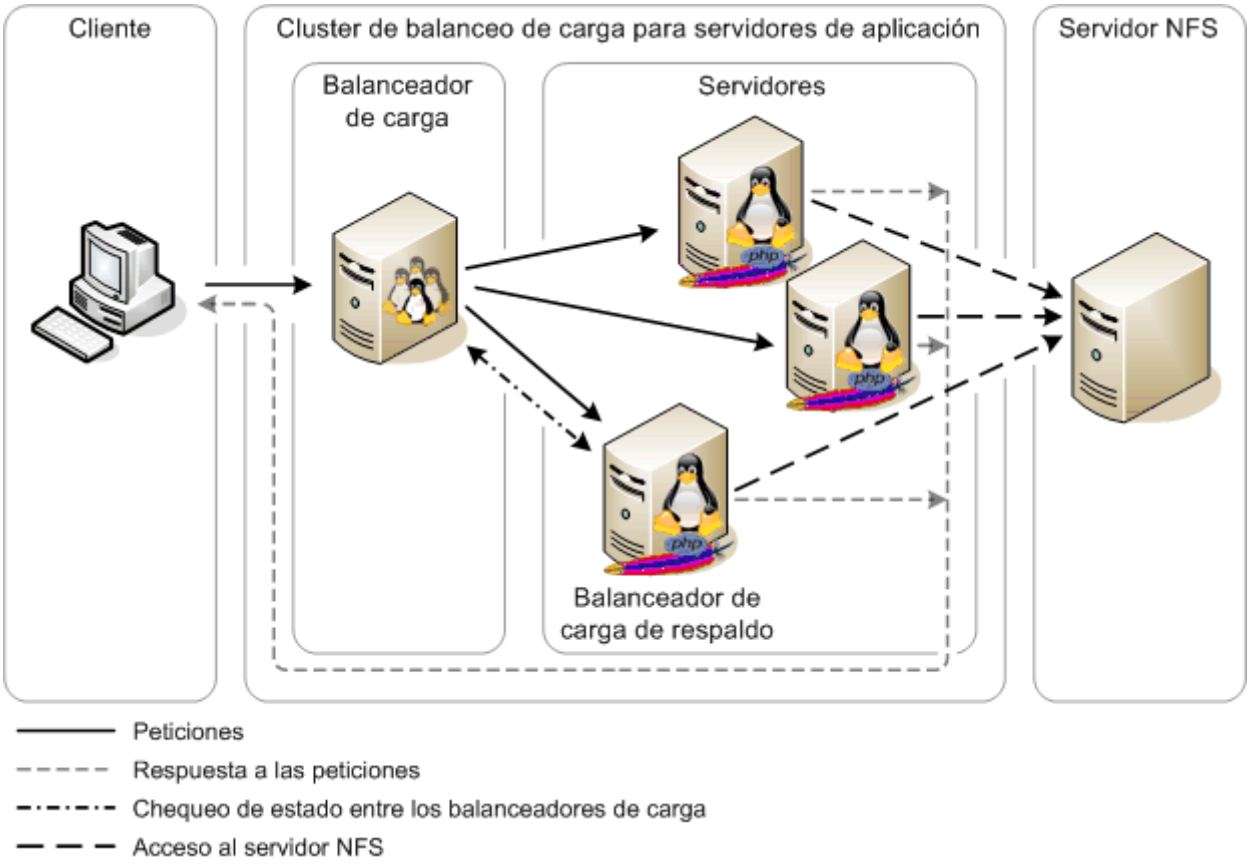

Figura 3.11. Estructura del cluster de balanceo de carga para EVA UCI en estado normal.

En el momento que el balanceador de carga primario quede fuera de servicio, como se muestra en la figura 3.12, el balanceador de carga de respaldo dejará de funcionar como servidor real y se convertirá en el balanceador de carga del sistema.

Una vez que el balanceador de carga primario este de alta nuevamente el balanceador de carga de respaldo vuelve a su estado de servidor real y balanceador de carga de respaldo.

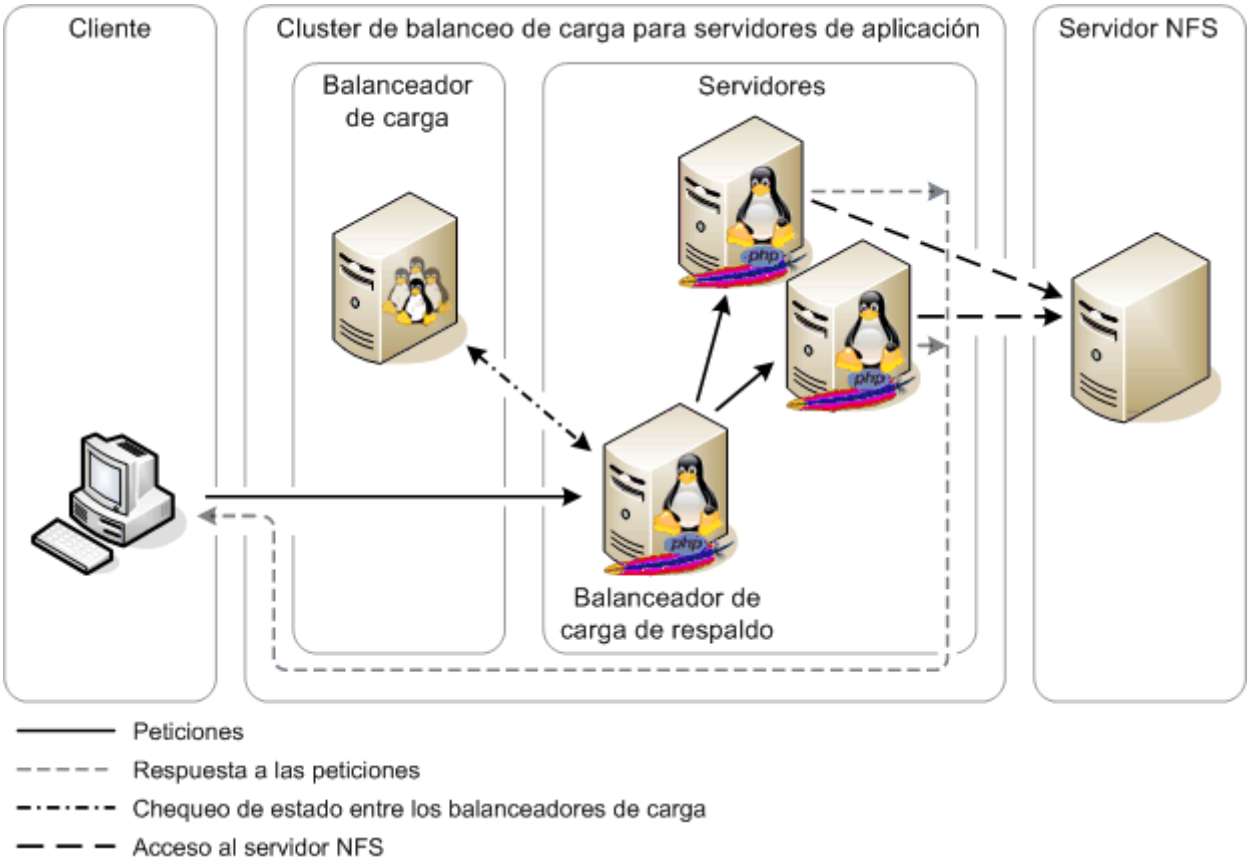

Figura 3.12. Estructura del cluster de balanceo de carga para EVA UCI cuando falla al balanceador de carga primario.

3.7. Conclusiones

El estudio de las características de cada una de las aplicaciones de muestra para realizar los diseños específicos de la propuesta de solución para cada una de ellas permitió concluir que:

1. Para las aplicaciones médicas que se despliegan en tres niveles existen tres variantes de la propuesta de solución: una para la capa de presentación, otra para la capa de lógica de negocio y una tercera que es la combinación de las anteriores.
2. Para las aplicaciones médicas que se despliegan en dos niveles solo existe una variante de la propuesta de solución.
3. El diseño del cluster de balanceo de carga para EVA UCI puede ser mejorado e incluir recuperación ante la caída del balanceador de carga.

CAPÍTULO 4. DISEÑO DE LAS PRUEBAS Y ANÁLISIS DE LOS RESULTADOS

Este capítulo está dedicado a la descripción y análisis de resultados de las pruebas de configuración y carga realizadas al diseño genérico y a cada una de las variantes de la propuesta de solución. Tiene como objetivo validar todos los diseños realizados.

4.1. Descripción del proceso de pruebas

El proceso de pruebas, que incluyo pruebas de configuración y carga, se realizó con el objetivo de validar las variantes de la propuesta de solución, mostrando que las aplicaciones funcionan correctamente sobre ellas y que su utilización permite disminuir los tiempos de respuesta del sistema.

Durante la realización de las pruebas de configuración se estudió el comportamiento de las funcionalidades de cada solución, para ver si se desempeñan como se previó en el diseño de la variante de la propuesta de solución. Las funcionalidades de la solución que se probaron fueron las que definen el comportamiento del cluster y pueden afectar el correcto funcionamiento de la aplicación.

Las pruebas de carga, para evaluar el rendimiento del sistema, se realizaron utilizando computadoras personales con características diferentes a las de los servidores sobre los que están o se van a desplegar las aplicaciones. Esto trae consigo que los resultados no reflejan la realidad exacta, pero si permiten apreciar las perspectivas de de la solución.

La métrica utilizada en las pruebas de carga para medir el rendimiento del sistema fue el tiempo de respuesta promedio y para medir el consumo de recursos en los servidores: % de uso del CPU, % de consumo de memoria RAM y % de uso de la red.

La generación de carga a los sistemas sobre los que se ejecuten las aplicaciones que se utilizaron en el proceso de pruebas se realizó mediante la navegación simultánea de un determinado número de usuarios por ellas. Para generar los usuarios concurrentes navegando por la aplicación se utilizó la herramienta JMeter. Esta herramienta permite grabar una navegación de un usuario por una aplicación y diseñar un plan de pruebas de con ella, para poder repetirla y simular el número de usuarios concurrentes que se desee.

Para grabar la navegación de un usuario por una aplicación se utiliza el componente HTTP Proxy Server que trae la herramienta JMeter. Este se configura para que actúe como proxy y a la vez grabe todas las peticiones que se realicen al sistema. Para que pueda grabar es necesario que el navegador que utilice el usuario esté configurado para utilizar este proxy.

Para repetir la navegación grabada es necesario añadir un elemento de configuración llamado HTTP Cookie Manager al plan de pruebas. Este elemento almacena las cookies que genera la aplicación y permite a la herramienta mantener las sesiones de usuario.

Las mediciones del tiempo de respuesta promedio de los sistemas se realizaron utilizando la herramienta JMeter y los planes de pruebas que se diseñaron para la generación de carga. A estos últimos se le añadió un elemento de reporte que recoge los tiempos de respuesta y las peticiones que conforman cada página se agruparon utilizando controladores de transacción. La agrupación de las peticiones se realiza con el propósito de conocer el tiempo que tarda el sistema en devolver una página completa.

Para efectuar las mediciones antes mencionadas, lo primero que se hizo fue generar la carga deseada simulando un determinado número de usuarios concurrentes y esperar que la aplicación web en cuestión se adaptara a ella. Después, desde una computadora diferente a las que están generando carga, se realizó la navegación de un usuario un total de diez veces para obtener el tiempo de respuesta promedio del sistema.

El consumo de recursos en los servidores web y en el balanceador de carga se midió utilizando la herramienta Sysstat. Al igual que en la medición de tiempo de respuesta del sistema, se esperó a que los servidores web se adaptaran a gestionar la carga que genera la aplicación sobre ellos antes de comenzar a medir. Para recolectar la información relativa al % de uso de CPU, consumo de memoria RAM y tráfico de red se ejecutó el comando “sa1” que trae la herramienta con los parámetros necesarios para que realizara una medición cada dos segundos hasta completar 200.

Después de terminada cada una de las pruebas se procesó la información recolectada para obtener los valores promedio del % de uso de CPU, % de consumo de memoria RAM y % de uso de la red en cada uno de los servidores. Para hacer esto se utilizó el comando “sar” perteneciente a la herramienta

Sysstat. Posteriormente, los valores obtenidos en cada servidor se promediaron para obtener el consumo de recursos promedio de los servidores web de manera general para cada prueba realizada.

4.2. Pruebas al diseño genérico de la solución

Todas las variantes de la propuesta de solución se basan en un diseño genérico del cluster de balanceo de carga que incluye recuperación ante fallas en los servidores web y ante la caída del balanceador de carga primario.

Para validar que este diseño se desempeña correctamente se realizaron pruebas de configuración a las principales funcionalidades del cluster. El listado de las pruebas realizadas y el propósito que cada una de estas se describe en la tabla 4.1. Para obtener más detalles sobre la realización de estas pruebas puede consultar los documentos de planificación y análisis de resultados de estas pruebas que se adjuntan en formato digital.

Tabla 4.1. Pruebas realizadas al diseño genérico del cluster de balanceo de carga

Referencia	Tipo	Propósito
GEN-CO-01	Configuración	Comprobar que el diseño genérico de la propuesta de solución realiza el balanceo de carga correctamente.
GEN-CO-02	Configuración	Comprobar que el diseño genérico de la propuesta de solución es capaz de recuperase ante fallos en los servidores web sin afectar el funcionamiento del sistema.
GEN-CO-03	Configuración	Comprobar que el diseño genérico de la propuesta de solución es capaz de recuperarse ante la caída del balanceador de carga primario.

Siguiendo las indicaciones del documento “Instalación y configuración de un cluster de balanceo de carga genérico para servidores web”, se realizó el despliegue de un cluster genérico que estaba compuesto por un balanceador de carga primario, uno de respaldo y dos servidores web.

Prueba de configuración GEN-CO-01

Para comprobar que el balanceo de carga se realiza efectivamente y según el algoritmo seleccionado fue necesario desplegar sobre el cluster varias páginas web que incluían en su diseño algunas imágenes. Se utilizó la herramienta JMeter para grabar una navegación para por las páginas web montadas y después de haber eliminado las entradas en el registro de accesos de cada servidor web, la navegación grabada se repitió diez veces simulando, en cada vez, cinco usuarios concurrentes. Como resultado de la navegación se realizaron 950 peticiones al balanceador de carga, 454 fueron atendidas por un servidor y 496 por el otro.

Prueba de configuración GEN-CO-02

El propósito de esta prueba fue comprobar que el balanceador de carga es capaz de detectar cuando un servidor web está fuera de servicio y que se autoconfigura para que esto no afecte el funcionamiento del sistema.

Para comprobar esto se comenzó a navegar por las páginas montadas y en medio de esta un servidor web fue desconectado de la red. El usuario que estaba navegando tuvo la necesidad de recargar una página pero pudo seguir sin más inconvenientes. Cuando se revisó la configuración del balanceador de carga, esta ya no permitía que se le reenviaran peticiones al servidor web desconectado.

Prueba de configuración GEN-CO-03

Antes de realizar esta prueba en el cluster se incluyó un sistema de archivos distribuido, con el servidor instalado junto a los balanceadores de carga, para comprobar que esta funcionalidad también se recupera ante la caída del balanceador de carga primario.

Para simular una falla en el balanceador de carga primario este se desconectó de la red. Observando el comportamiento del balanceador de carga de respaldo se pudo ver como detectó que el balanceador de carga primario estaba fuera de servicio. Como respuesta a esto, configuró la VIP y la dirección IP por la que se accede al sistema de archivos distribuido, levanto el servicio de balanceo de carga y mandó a reiniciarse a los servidores web.

Cuando se conectó la red al balanceador de primario y el de respaldo lo detectó mandó a reiniciarse nuevamente a los servidores web, paró el servicio de balanceo de carga y desconfiguró la dirección IP por la que se accede al sistema de archivos distribuido y la VIP. El balanceador de carga primario tomó su lugar y todo siguió funcionando como antes de simular la falla.

Análisis de los resultados

Durante las pruebas de configuración realizadas al diseño genérico del cluster de balanceo de carga se probaron las funcionalidades: balanceo de carga, recuperación ante fallos de los servidores web y recuperación del cluster ante la caída del balanceador de carga primario. Como el comportamiento obtenido para cada una de las funcionalidades probadas fue satisfactorio se puede concluir que: el diseño del cluster de balanceo de carga genérico está validado funcionalmente y que en las variantes de la propuesta de solución, que se basan en este, solo es necesario probar las funcionalidades específicas.

4.3. Pruebas a la primera variante de la propuesta de solución para la aplicación RPS

Para validar la variante de la propuesta de solución para la aplicación Registro de personal de la salud que implementa un cluster de balanceo de carga para la capa de presentación, se realizaron pruebas de configuración y de carga. En la tabla 4.2 se especifican las pruebas realizadas y el propósito de cada una.

Tabla 4.2. Pruebas realizadas a la solución propuesta para la capa de presentación de la aplicación Registro de personal de la salud.

Referencia	Tipo	Propósito
RPS_CP-CO-04	Configuración	Comprobar que el sistema de archivos distribuido que incluye la propuesta de solución para la capa de presentación de la aplicación RPS funciona como se previó.
RPS_CP-CO-05	Configuración	Comprobar que la aplicación se comporta de la misma manera cuando su capa de presentación se ejecuta sobre la variante de la propuesta de solución diseñada para ella y sobre el despliegue original.
RPS_CP-CA-06	Carga	Encontrar el número de usuarios concurrentes que al navegar por la aplicación provocan que el servidor sobre el que se ejecuta la capa de presentación llegue al límite en el consumo de alguno de sus recursos.
RPS_CP-CA-07	Carga	Mostrar como disminuye el tiempo de respuesta del sistema cuando se aumenta el número de servidores web en el cluster de balanceo de carga sobre el que se ejecuta la capa de presentación de la aplicación RPS.

Los documentos de planificación y análisis de resultados de estas pruebas se adjuntan en formato digital y pueden ser consultados para obtener más detalles sobre ellas.

Para la realización de las pruebas la capa de presentación de la aplicación se desplegó sobre una configuración básica del cluster de balanceo de carga, compuesta por dos balanceadores de carga y dos servidores capa de presentación. La capa de lógica de negocio y la capa de datos se desplegaron cada una sobre servidores independientes.

La instalación de los servidores de capa de datos, lógica de negocio y capa de presentación se realizó como se describe en el documento de despliegue de la aplicación. La instalación y configuración del cluster de balanceo de carga se realizó como se describe en el documento “Cluster de balanceo de carga para la capa de presentación de la aplicación Registro de personal de la salud”.

Prueba de configuración RPS_CP-CO-04

Esta prueba se realizó para probar que el almacenamiento en el sistema de archivos distribuido de los ficheros de sesión PHP funciona correctamente y que esos ficheros pueden ser utilizados por todos los servidores de capa de presentación sin afectar el funcionamiento de la misma.

Para ejecutar la prueba se configuró el balanceador de carga para que solo reenviara las peticiones a uno de los servidores de capa de presentación. Con el cluster configurado de esta manera un usuario navegó por la aplicación. Después que pasó la autenticación, el balanceador de carga se reconfiguró para que reenviara las peticiones al otro servidor de capa de presentación y el usuario siguió navegando por la aplicación sin notar ningún cambio. También se comprobó que el fichero de sesión PHP correspondiente a la sesión del usuario en la aplicación estaba en el sistema de archivos distribuido.

Prueba de configuración RPS_CP-CO-05

Para ejecutar esta prueba, además del despliegue existente de la aplicación, fue necesario realizar el despliegue original de la misma. La prueba consistió en realizar, con el mismo usuario del sistema, una navegación simultánea por ambos despliegues y se comprobó que el comportamiento, en cuanto a funcionamiento, es idéntico.

La navegación se realizó por las páginas que implementan las siguientes funcionalidades:

- Autenticar usuario
- Buscar personal de la salud
- Insertar personal de la salud
- Generar reportes.

Prueba de carga RPS_CP-CA-06

Para ejecutar esta prueba la capa de presentación se mantuvo desplegada sobre un servidor como en el despliegue original. Además se garantizó que la capacidad de procesamiento de las capas de lógica de negocio y de datos fuera suficiente para que las variaciones en los resultados solo dependieran de la capa de presentación.

La generación de carga necesaria para la prueba se realizó desde tres computadoras, en las que solo se ejecutaba la herramienta JMeter y plan de pruebas utilizado simulaba la navegación de un usuario por 26 páginas de la aplicación.

Como resultado se obtuvo el tiempo de respuesta promedio del sistema para los niveles de concurrencia de cinco, diez y quince usuarios y el consumo de recursos en el servidor de capa de presentación para cada uno de ellos. Estos valores se muestran en la tabla 4.3.

Tabla 4.3. Tiempos de respuestas promedio del sistema y valores del consumo de recursos en el servidor de capa de presentación obtenidos durante la ejecución de la prueba RPS_CP-CA-06.

	Niveles de concurrencia		
	5 usuarios	**10 usuarios**	**15 usuarios**
Tiempo de respuesta promedio (segundos)	0.52	0.91	1.31
% de uso de CPU	96.89	99.83	99.89
% de consumo de memoria RAM	8.22	11.06	13.24
% de uso de la red	13.50	13.75	13.71

Analizando estos resultados, se puede observar, que cuando navegan por la aplicación 15 usuarios concurrentes, el uso de CPU en el servidor de capa de presentación alcanza valores muy próximos al límite. Basándose en esto se decidió que 15 usuarios concurrentes sería el nivel de concurrencia máximo que iba a soportar un servidor de capa de presentación durante la realización de las pruebas.

Prueba de carga RPS_CP-CA-07

Para efectuar esta prueba la aplicación se sometió a un nivel de concurrencia de 15 usuarios mientras se ejecutaba sobre configuraciones del cluster de dos, tres y cuatro servidores. Teniendo en cuenta las mismas consideraciones que en la prueba RPS_CP-CA-06, se garantizó que las variaciones en los resultados solo dependieran de la capa de presentación. La generación de carga también se realizó de la misma forma que en la prueba antes mencionada.

Además del tiempo de respuesta promedio del sistema, para cada una de las configuraciones se recogió el promedio del % de uso de CPU, del % de consumo de memoria RAM y del % de uso de la red de los servidores de capa de presentación. Todos los datos recogidos se muestran en la tabla 4.4.

Tabla 4.4. Tiempos de respuestas promedio del sistema y valores del consumo de recursos en los servidores de capa de presentación obtenidos durante la ejecución de la prueba RPS_CP-CA-07.

	Configuraciones del cluster		
	2 servidores	**3 servidores**	**4 servidores**
Tiempo de respuesta promedio (segundos)	0.64	0.41	0.34
% de uso de CPU	98.19	88.78	78.20
% de consumo de memoria RAM	11.45	9.93	8.78
% de uso de la red	18.46	17.34	15.82

Durante esta prueba también se recogió los valores del consumo de recursos promedio en el balanceador de carga para cada una de las configuraciones del cluster, estos valores se pueden observar en la tabla 4.5.

Tabla 4.5. Valores del consumo de recursos en el balanceador de carga para las diferentes configuraciones del cluster obtenidos durante la ejecución de la prueba RPS_CP-CA-07.

	Configuraciones del cluster		
	2 servidores	**3 servidores**	**4 servidores**
% de uso de CPU	2.30	3.00	3.59
% de consumo de memoria RAM	18.83	18.83	18.81
% de uso de la red	13.88	19.55	23.84

Análisis de resultados

En el cluster de balanceo de carga para la capa de presentación de la aplicación Registro de personal de la salud se probaron las funcionalidades, almacenamiento en el sistema de archivos distribuido de los ficheros de sesión PHP y el funcionamiento correcto de la aplicación. Ambas funcionalidades tuvieron un comportamiento satisfactorio.

En la gráfica de la figura 4.1 se puede apreciar como, para un nivel de concurrencia de 15 usuarios, el tiempo de respuesta promedio del sistema disminuye con el aumento del número servidores en la configuración del cluster para la capa de presentación. También se puede apreciar la diferencia entre estos y el que se obtuvo cuando la capa de presentación se ejecutaba sobre un solo servidor en el despliegue original.

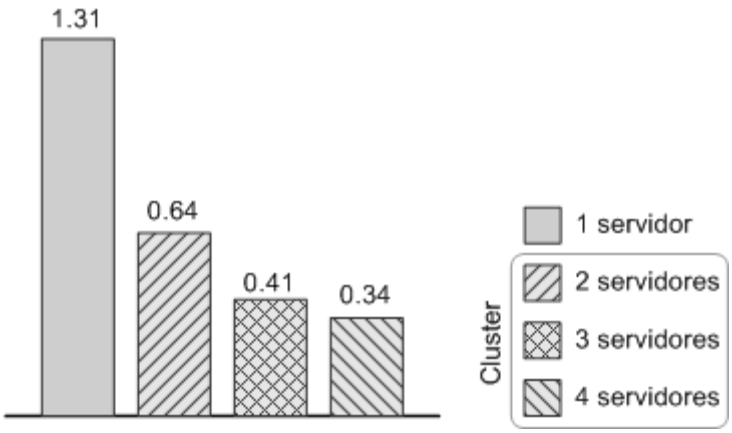

Figura 4.1. Tiempos de respuesta promedio del sistema, en segundos, para un servidor de capa de presentación y las diferentes configuraciones del cluster para la misma capa.

Analizando la gráfica del consumo de recursos promedio para el despliegue original de la capa de presentación y las diferentes configuraciones del cluster, figura 4.2, se observa como la disminución de los tiempos de respuesta está soportada por una mejora en el consumo de los recursos.

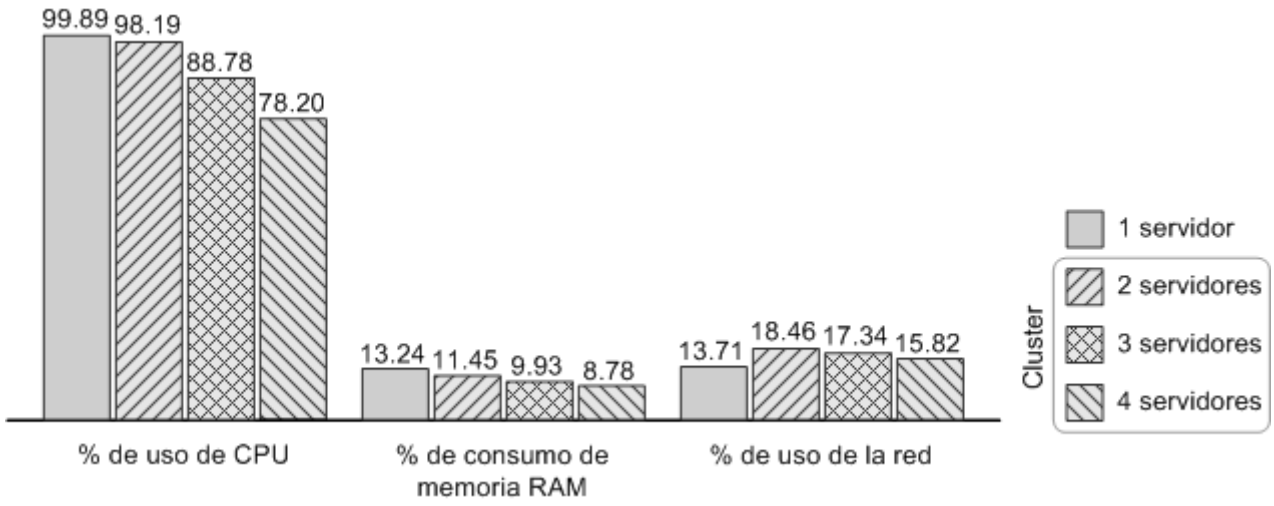

Figura 4.2. Consumo de recursos en los servidores de capa de presentación.

El consumo de recursos en el balanceador de carga no alcanzó valores preocupantes, pero es necesario analizarlo porque esto es lo que normalmente frena el escalamiento de la solución. Una forma de conocer cual recurso se agotará primero es analizar la tendencia del consumo de cada uno de estos.

Utilizando la función “ESTIMACION.LINEAL” de Microsoft Excel, que calcula la línea que mejor se ajuste a los datos utilizando el método de "mínimos cuadrados", se pudo obtener una función lineal que describe la tendencia en el consumo de cada uno de los recursos en el balanceador de carga. La graficación de estas funciones se muestra en la figura 4.3.

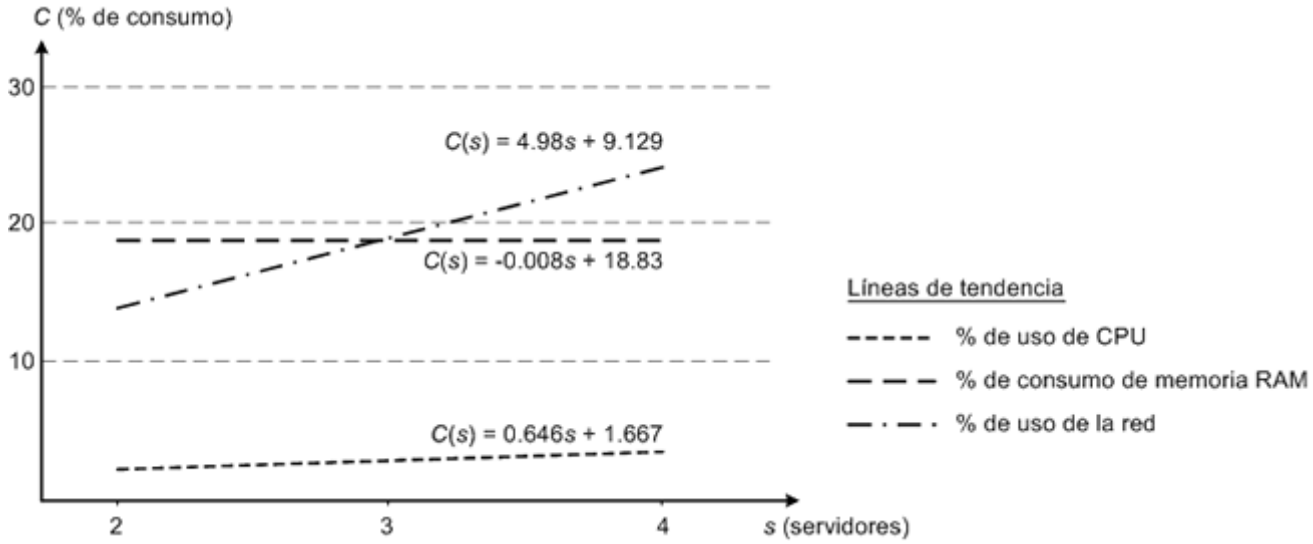

Figura 4.3. Funciones que describen la tendencia del consumo de recursos en el balanceador de carga.

La función que crece más rápido es $C(s) = 4.98s + 9.129$ y representa el % de uso de la red. Haciendo una extrapolación de esta se pudo conocer que 18 servidores de capa de presentación es el escalamiento máximo que soporta el balanceador de carga del cluster utilizado para la realización de las prueba a esta variante de la propuesta de solución.

Las pruebas de configuración y carga realizadas a la variante de la propuesta de solución para la capa de presentación de la aplicación Registro de personal de la salud mostraron que su diseño no afecta el funcionamiento de la aplicación y logra mejorar los tiempos de respuesta del sistema. Basado en esto se puede concluir que el diseño es valido para la capa de presentación de esta aplicación y por ende para las del grupo que ella representa.

4.4. Pruebas a la segunda variante de la propuesta de solución para la aplicación RPS

En la tabla 4.6 están registradas las pruebas de configuración y carga realizadas para validar la segunda variante de la propuesta de solución para la aplicación Registro de personal de la salud. Para obtener más detalles sobre estas pruebas puede revisar los documentos de planificación y análisis de resultados que se adjuntan en formato digital.

Tabla 4.6. Pruebas realizadas a la variante de la propuesta de solución para la capa de lógica de negocio de la aplicación Registro de personal de la salud

Referencia	Tipo	Propósito
RPS_LN-CO-08	Configuración	Comprobar que el sistema de archivos distribuido de la variante de la propuesta de solución para la capa de lógica de negocio de la aplicación funciona correctamente.
RPS_LN-CO-09	Configuración	Comprobar que la aplicación se comporta de la misma manera cuando su capa de lógica de negocio se ejecuta sobre la variante de la propuesta de solución diseñada para ella y sobre el despliegue original.
RPS_LN-CA-10	Carga	Encontrar el número de usuarios concurrentes que al navegar por la aplicación provocan que el servidor sobre el que se ejecuta la capa de lógica de negocio llegue al límite en el consumo de alguno de sus recursos.
RPS_LN-CA-11	Carga	Mostrar como disminuye el tiempo de respuesta del sistema cuando se aumenta el número de servidores web en el cluster de balanceo de carga sobre el que se ejecuta la capa de lógica de negocio de la aplicación.

Para la realización de las pruebas la capa de lógica de negocio de la aplicación se desplegó sobre una configuración básica del cluster de balanceo de carga, compuesta por dos balanceadores de carga y dos servidores de lógica de negocio. La capa de presentación y la capa de datos se desplegaron cada una sobre servidores independientes.

La instalación de los servidores de capa de datos, lógica de negocio y capa de presentación se realizó como se describe en el documento de despliegue de la aplicación. La instalación y configuración del cluster de balanceo de carga se realizó como se describe en el documento “Cluster de balanceo de carga para la capa de lógica de negocio de la aplicación Registro de personal de la salud”.

Prueba de configuración RPS_LN-CO-08

Para probar que el almacenamiento en el sistema de archivos distribuido de los reportes en formato PDF y EXCEL funciona correctamente y que esos ficheros pueden ser accedidos por todos los servidores de lógica de negocio, se configuró el balanceador de carga para que solo reenviara las peticiones a uno de los servidores del cluster. Con el cluster configurado de esta manera un usuario, que estaba navegando por la aplicación, mandó a generar un reporte. Después que se generó el reporte, el balanceador de carga se reconfiguró para que reenviara las peticiones al otro servidor de lógica de negocio y el usuario pudo descargar el reporte satisfactoriamente. También se comprobó que el reporte fue almacenado en el sistema de archivos distribuido.

Prueba de configuración RPS_LN-CO-09

Para realizar esta prueba, además del despliegue existente de la aplicación con la capa de lógica de negocio ejecutándose sobre un cluster de balanceo de carga, se utilizó el despliegue original de la aplicación que se realizó para la prueba RPS_CP-CO-05. La prueba fue igual que la antes mencionada y permitió comprobar el comportamiento de la aplicación sobre ambos despliegues es el mismo.

Prueba de carga RPS_LN-CA-10

Para ejecutar esta prueba la capa de lógica de negocio se mantuvo sobre un servidor como en el despliegue original de la aplicación. Las capas de presentación y de datos se desplegaron sobre configuraciones que garantizaban que la capacidad de procesamiento fuera suficiente para que las variaciones en los tiempos de respuesta solo dependieran de la capa de lógica de negocio. La generación de carga se realizó de la misma forma que en la prueba RPS_CP-CA-06.

Como resultado de la pruebas se obtuvo el tiempo de respuesta promedio del sistema para los niveles de concurrencia de 10, 20 y 25 usuarios y el consumo de recursos en el servidor de lógica de negocio para cada uno de ellos. Estos valores se muestran en la tabla 4.7.

Tabla 4.7. Tiempos de respuestas promedio del sistema y valores del consumo de recursos en el servidor de lógica de negocio obtenidos durante la ejecución de la prueba RPS_LN-CA-10.

	Niveles de concurrencia		
	15 usuarios	**20 usuarios**	**25 usuarios**
Tiempo de respuesta promedio (segundos)	0.66	0.81	0.98
% de uso de CPU	97.36	99.33	99.6
% de consumo de memoria RAM	13.71	15.88	18.4
% de uso de la red	2.96	2.99	3.02

Analizando estos resultados, se pudo apreciar, que cuando navegan por la aplicación 25 usuarios concurrentes, el uso de CPU en el servidor de capa de presentación alcanza valores muy próximos al límite. Basándose en esto se decidió que este sería el nivel de concurrencia máximo que iba a soportar un servidor de lógica de negocio durante la realización de las pruebas.

Prueba de carga RPS_LN-CA-11

Para efectuar esta prueba la aplicación se sometió a un nivel de concurrencia de 25 usuarios mientras se ejecutaba sobre configuraciones del cluster de dos, tres y cuatro servidores. Se tuvieron en cuenta las mismas consideraciones que para la prueba RPS_LN-CA-10 para garantizar que las variaciones en los resultados solo dependieran de la capa de lógica de negocio y la generación de carga se realizó de la misma forma que en la prueba RPS_CP-CA-06.

El propósito de la prueba fue obtener el tiempo de respuesta promedio del sistema para cada una de las configuraciones del cluster para la capa de lógica de negocio, pero además se recogió el promedio del % de uso de CPU, del % de consumo de memoria RAM y del % de uso de la red de los servidores web que lo integran. Todos los datos recogidos se muestran en la tabla 4.8.

Tabla 4.8. Tiempos de respuestas promedio del sistema y valores del consumo de recursos en los servidores de lógica de negocio obtenidos durante la ejecución de la prueba RPS_LN-CA-11.

	Configuraciones del cluster		
	2 servidores	**3 servidores**	**4 servidores**
Tiempo de respuesta promedio (segundos)	0.54	0.41	0.38
% de uso de CPU	81.53	55.67	40.57
% de consumo de memoria RAM	10.48	7.91	6.92
% de uso de la red	2.58	1.94	1.53

También se recogió los valores del consumo de recursos promedio en el balanceador de carga para las diferentes configuraciones del cluster que fueron probadas, estos valores se pueden observar en la tabla 4.9.

Tabla 4.9. Valores del consumo de recursos en el balanceador de carga para las diferentes configuraciones del cluster obtenidos durante la ejecución de la prueba RPS_LN-CA-11.

	Configuraciones del cluster		
	2 servidores	**3 servidores**	**4 servidores**
% de uso de CPU	0.62	0.67	0.70
% de consumo de memoria RAM	8.70	8.38	8.02
% de uso de la red	0.98	1.12	1.20

Análisis de resultados

Las funcionalidades, almacenamiento en el sistema de archivos distribuido de los reportes y funcionamiento correcto de la aplicación, del cluster de balanceo de carga para la capa de lógica de negocio de la aplicación Registro de personal de la salud, que fueron probadas durante las pruebas de configuración tuvieron un comportamiento satisfactorio.

En la gráfica de la figura 4.4 se puede apreciar como, para un nivel de concurrencia de 25 usuarios, el tiempo de respuesta promedio del sistema disminuye con el aumento del número servidores en la configuración del cluster. También es apreciable la diferencia entre estos y el que se obtuvo cuando la capa de lógica de negocio se ejecutaba sobre un servidor simulando el despliegue original.

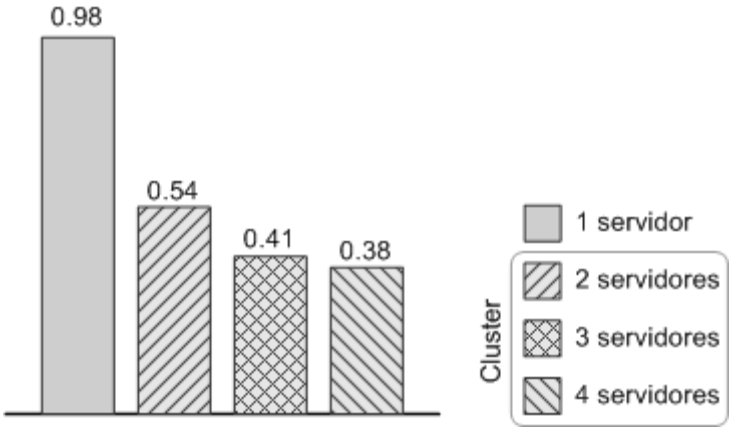

Figura 4.4. Tiempos de respuesta promedio del sistema, en segundos, para un servidor y las diferentes conjuraciones del cluster.

Igual al comportamiento obtenido durante la realización de las pruebas de carga a la primera variante de la propuesta de solución para esta aplicación, la mejora de los tiempos de respuesta del sistema está soportada por una la mejora en el consumo de recursos en los servidores de lógica de negocio, esto se muestra en la gráfica de la figura 4.5.

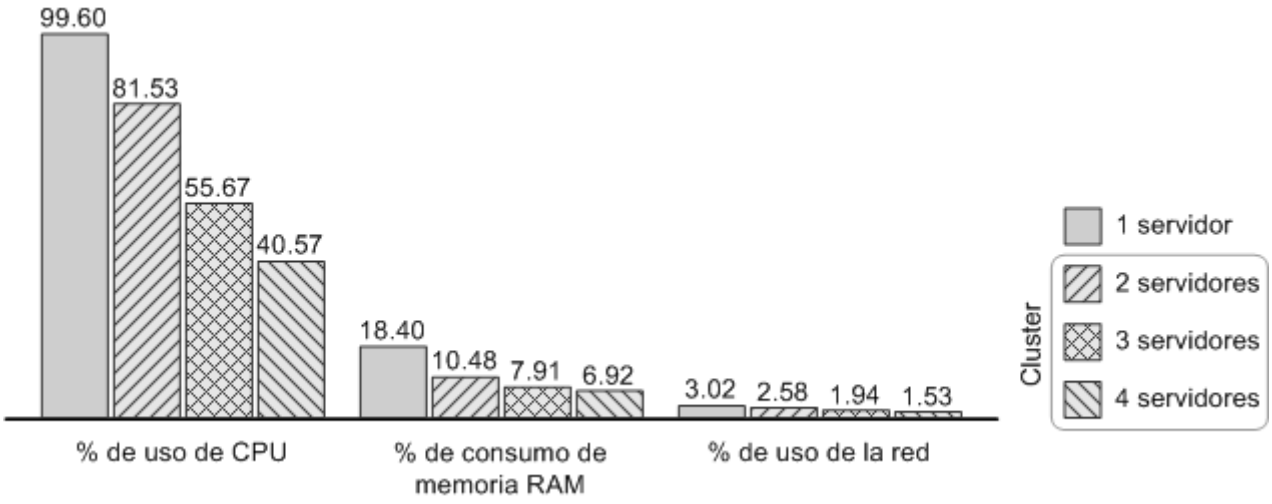

Figura 4.5. Consumo de recursos en los servidores de lógica de negocio para las diferentes configuraciones del cluster y un servidor independiente.

Para esta variante de la propuesta de solución se hizo el mismo análisis del consumo de recursos en el balanceador de carga que se realizó en el epígrafe anterior. Este permitió determinar que la configuración del cluster para la capa de lógica de negocio utilizado durante la realización de las pruebas, puede soportar más de 100 servidores web.

Las pruebas de configuración y carga realizadas a la variante de la propuesta de solución para la capa de lógica de negocio de la aplicación Registro de personal de la salud mostraron que su diseño no afecta el funcionamiento de la aplicación y logra mejorar los tiempos de respuesta del sistema. Basado en esto se puede concluir que el diseño es válido para sustituir los servidores web sobre los que se ejecutan las capas de lógica de negocio de este tipo de aplicaciones.

4.5. Pruebas a la variante de la propuesta de solución para la aplicación Balance Material

Igual que para validar las variantes de solución anteriores, se realizaron pruebas de configuración y de carga para aprobar el diseño del cluster realizado para ejecutar las capas de presentación y lógica de negocio de la aplicación Balance Material. En la tabla 4.10 están registradas todas las pruebas realizadas y el propósito de cada una. Los documentos de planificación y análisis de resultados de estas pruebas se adjuntan en formato digital.

Tabla 4.10. Pruebas realizadas a la variante de la propuesta de solución para las capas de presentación y negocio de la aplicación Balance Material.

Referencia	Tipo	Propósito
BM-CO-12	Configuración	Comprobar que el sistema de archivos distribuido de la variante de la propuesta de solución para la aplicación funciona correctamente.
BM-CO-13	Configuración	Comprobar que la aplicación se comporta de la misma manera al ejecutarse sobre la variante de la propuesta de solución diseñada para ella y sobre el despliegue original.
BM-CA-14	Carga	Encontrar el número de usuarios concurrentes que al navegar por la aplicación provocan que el servidor sobre el que esta se ejecuta, llegue al límite en el consumo de alguno de sus recursos.
BM-CA-15	Carga	Mostrar como al aumentar el número de servidores de aplicación en el cluster de balanceo de carga, manteniendo el nivel de concurrencia constante, disminuye el tiempo de respuesta promedio del sistema.

Para la realización de las pruebas las capas de presentación y negocio de la aplicación se desplegaron sobre una configuración básica del cluster de balanceo de carga, compuesta por dos balanceadores de carga y dos servidores aplicación. La capa de datos se desplegó sobre un servidor independiente.

La instalación de los servidores de aplicación y capa de datos se realizó como se describe en el documento de despliegue de la aplicación. La instalación y configuración del cluster de balanceo de carga se realizó como se describe en el documento “Cluster de balanceo de carga para la aplicación Balance Material”.

Prueba de configuración BM-CO-12

Esta prueba se realizó igual que la RPS_CP-CO-04. Después de realizarla se pudo comprobar que el almacenamiento en el sistema de archivos distribuido de las sesiones PHP funciona correctamente para esta variante de la propuesta de solución.

Prueba de configuración BM-CO-13

Para ejecutar esta prueba, además del despliegue existente de la aplicación, fue necesario realizar el despliegue original de la misma. La prueba consistió en realizar, con el mismo usuario del sistema, una navegación simultánea por ambos despliegues de la aplicación y se comprobó que el comportamiento de las funcionalidades de la aplicación se mantiene. La navegación se realizó por las páginas que implementan las siguientes funcionalidades:

- Autenticar usuario.
- Gestionar demanda.
- Gestionar planificación.
- Verificar existencia de materiales.
- Buscar materiales.

Prueba de carga BM-CA-14

Para ejecutar esta prueba las capas de presentación y negocio de la aplicación se ejecutaron sobre un servidor como en el despliegue original. A la capa de datos se le garantizó suficiente capacidad de procesamiento para que los resultados de las pruebas solo dependieran del servidor sobre el que se ejecutan las otras capas de la aplicación.

Como resultado de la prueba se obtuvo el tiempo de respuesta promedio del sistema para los niveles de concurrencia de cinco, diez y quince usuarios y el consumo de recursos en el servidor de aplicación para cada uno de ellos. Estos valores se muestran en la tabla 4.11.

Tabla 4.11. Tiempos de respuestas promedio del sistema y valores del consumo de recursos en el servidor de aplicación obtenidos durante la ejecución de la prueba BM-CA-14.

	Niveles de concurrencia		
	5 usuarios	**10 usuarios**	**15 usuarios**
Tiempo de respuesta promedio (segundos)	0.37	0.67	0.94
% de uso de CPU	95.47	99.29	99.86
% de consumo de memoria RAM	9.18	12.60	14.30
% de uso de la red	18.78	18.96	19.12

Analizando estos resultados, se pudo apreciar, que cuando navegan por la aplicación 15 usuarios concurrentes, el uso de CPU en el servidor de aplicación alcanza valores muy próximos al límite. Basándose en esto se decidió que 15 usuarios sería el nivel de concurrencia máximo que iba a soportar un servidor de aplicación durante la realización de las pruebas.

Prueba de carga BM-CA-15

Para efectuar esta prueba la aplicación se sometió a un nivel de concurrencia de 15 usuarios mientras se ejecutaba sobre configuraciones del cluster de dos, tres y cuatro servidores. Al igual que en la prueba anterior, a la capa de datos se le garantizó suficiente capacidad de procesamiento para que los resultados de las pruebas solo dependieran de las configuraciones del cluster sobre el que se ejecutan las otras capas de la aplicación.

Junto al tiempo de respuesta promedio del sistema, para cada una de las configuraciones se recogió el promedio del % de uso de CPU, del % de consumo de memoria RAM y del % de uso de la red en los servidores de aplicación. Todos los datos recogidos se muestran en la tabla 4.12.

Tabla 4.12. Tiempos de respuestas promedio del sistema y valores del consumo de recursos en los servidores de aplicación obtenidos durante la ejecución de la prueba BM-CA-15.

	Configuraciones del cluster		
	2 servidores	**3 servidores**	**4 servidores**
Tiempo de respuesta promedio (segundos)	0.47	0.31	0.23
% de uso de CPU	99.77	97.89	93.28
% de consumo de memoria RAM	12.52	10.83	9.64
% de uso de la red	19.67	19.57	19.17

Además se recogió los valores del consumo de recursos promedio en el balanceador de carga para cada una de las configuraciones, estos valores se pueden observar en la tabla 4.13.

Tabla 4.13. Valores del consumo de recursos en el balanceador de carga para las diferentes configuraciones del cluster obtenidos durante la ejecución de la prueba BM-CA-15.

	Configuraciones del cluster		
	2 servidores	**3 servidores**	**4 servidores**
% de uso de CPU	2.30	3.23	3.88
% de consumo de memoria RAM	18.83	18.83	18.83
% de uso de la red	6.92	10.52	13.94

Análisis de resultados

Las funcionalidades del cluster: almacenamiento en el sistema de archivos distribuido de las sesiones PHP y funcionamiento correcto de la aplicación, probadas durante la realización de las pruebas de configuración, demostraron que el diseño de la variante de la propuesta de solución para la aplicación Balance Material no afecta el funcionamiento de la misma.

Como se puede apreciar en la gráfica de la figura 4.6, para un nivel de concurrencia de 15 usuarios, el tiempo de respuesta promedio del sistema disminuye con el aumento del número servidores en la configuración del cluster. Además existe una gran diferencia entre estos y el que se obtuvo cuando las capas de presentación y lógica de negocio se ejecutaban sobre un solo servidor.

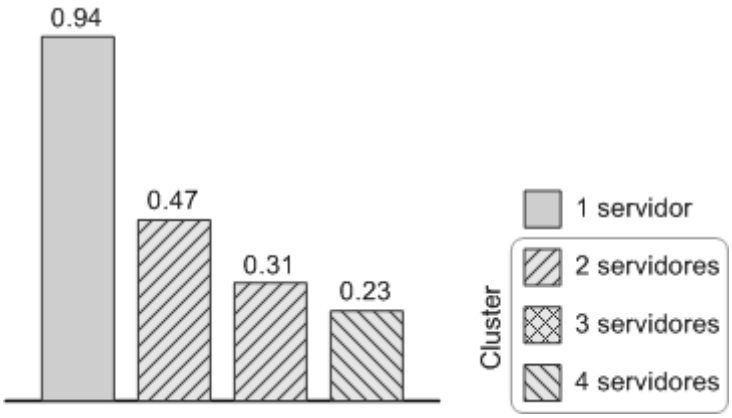

Figura 4.6. Tiempos de respuesta promedio del sistema, en segundos, para un servidor y las diferentes conjuraciones del cluster.

Al igual que lo ocurrido con las variantes de la propuesta de solución probadas anteriormente, esta mejora en los tiempos de respuesta está soportada por una mejora en el consumo de los recursos, como se puede observar en la gráfica de la figura 4.7.

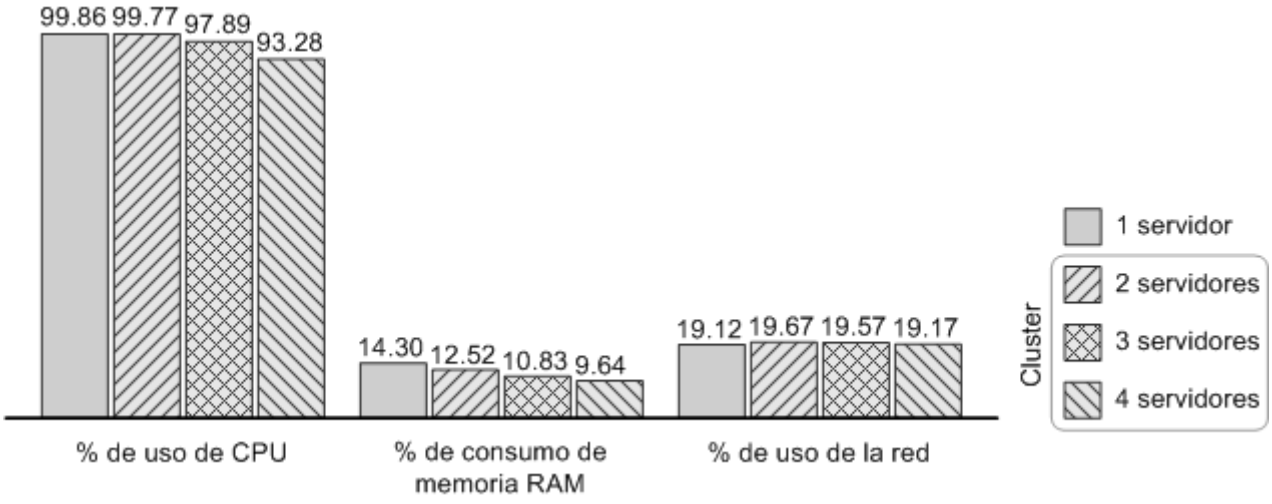

Figura 4.7. Consumo de recursos en los servidores de capa de aplicación.

El consumo de recursos en el balanceador de carga no alcanzó valores preocupantes, pero se analizó igual que para las variantes de la propuesta de solución anteriores para poder estimar el escalamiento de esta variante. Como resultado se obtuvo que un balanceador de carga, con las características del hardware que se utilizó para las pruebas, puede gestionar aproximadamente 27 servidores web para esta aplicación.

Del análisis conjunto de los resultados de las pruebas de configuración y carga, realizadas a la variante de la propuesta de solución para las capas de presentación y negocio de la aplicación Balance Material, se puede concluir que el diseño de la solución es valido y puede ser utilizado en las aplicaciones similares a ella.

4.6. Pruebas a la primera variante de la propuesta de solución para la aplicación RCD

Como se aclaró en el epígrafe 2.1, la aplicación Registro centralizado de donantes no gestiona altos niveles de concurrencia. Pero como tiene las características de un grupo de aplicaciones que serán desarrolladas en un futuro próximo, las cuales se pronostica que si van a gestionar altos niveles de concurrencia, se diseñaron variantes de la propuesta de solución para ella.

Para esta variante de la propuesta de solución no se realizaron pruebas de carga, esto trae como consecuencia que la validación es parcial.

Las pruebas de carga no se realizan solo porque la aplicación no gestiona altos niveles de concurrencia. Además, las características del diseño de la variante de solución para la capa de presentación introducen requerimientos que no se pueden cumplir en el entorno de pruebas.

Para generar carga a la variante de la solución en cuestión, las conexiones que realiza cada usuario deben originarse siempre desde la misma dirección IP y esta debe ser diferente para cada uno. Esto es imposible obtenerlo en el entorno de pruebas actual con las herramientas que se disponen.

Aunque no se pudo probar, se supone que con la utilización de esta variante de la propuesta de solución se logra disminuir los tiempos de respuesta del sistema. Como todas las variantes de la propuesta de solución son especificaciones de un diseño genérico, la suposición la soporta los resultados obtenidos en las pruebas de carga realizadas a las otras variantes de de la propuesta de solución.

A esta variante de la propuesta de solución solo fue necesario realizarle una prueba de configuración. La referencia de la prueba es RCD_CP-CO-16 y tuvo como propósito comprobar que la aplicación Registro centralizado de donantes se comporta de la misma manera cuando su capa de presentación se ejecuta sobre la variante de la propuesta de solución diseñada para ella y sobre el despliegue original.

Para hacer esta prueba fue necesario realizar dos despliegues de la aplicación, uno con la capa de presentación sobre la variante de la solución diseñada para ella y otra con la aplicación ejecutándose sobre el despliegue original.

El cluster sobre el que se ejecutó la capa de presentación estaba compuesto por un balanceador de carga primario, uno de respaldo y dos servidores de capa de presentación. La instalación y configuración de los servidores se realizó según el documento de despliegue de la aplicación y del cluster según el documento “Instalación y configuración del cluster de balanceo de carga para la capa de presentación de la aplicación Registro centralizado de donantes”

La prueba consistió en realizar, con el mismo usuario del sistema, una navegación simultánea por ambos despliegues de la aplicación y se comprobó que el comportamiento de las funcionalidades de la aplicación se mantiene. La navegación se realizó por las páginas que implementan las siguientes funcionalidades:

- Autenticar usuario
- Buscar donantes
- Mostrar datos generares

La realización de esta prueba permitió validar el funcionamiento correcto de la aplicación sobre la variante de la propuesta de solución diseñada para ella.

4.7. Pruebas a la segunda variante de la propuesta de solución para la aplicación RCD

A esta segunda variante de la propuesta de solución para la aplicación Registro centralizado de donantes, igual que a la primera, no se le pudo hacer pruebas de carga y solo se le realizó una prueba de configuración.

Las pruebas de carga no se realizaron debido a las características de la aplicación, que impiden obtener una capacidad de procesamiento suficiente en las capas de presentación y datos para que las variaciones en el rendimiento del sistema solo dependan de la capa de lógica de negocio. La capacidad de procesamiento requerida es imposible de obtener debido a que el *hardware* del que se dispone es insuficiente.

La referencia de la prueba de configuración que se realizó es RCD_LN-CO-17 y tuvo como propósito comprobar que la aplicación Registro centralizado de donantes se comporta de la misma manera cuando su capa de lógica de negocio se ejecuta sobre la variante de la propuesta de solución diseñada para ella y sobre el despliegue original.

Para hacer esta prueba fue necesario realizar dos despliegues de la aplicación, uno con la capa de lógica de negocio sobre la variante de la solución diseñada para ella y otra con la aplicación ejecutándose sobre el despliegue original.

El cluster sobre el que se ejecutó la capa en cuestión estaba compuesto por un balanceador de carga primario, uno de respaldo y dos servidores de lógica de negocio. La instalación y configuración de los servidores se realizó según el documento de despliegue de la aplicación y del cluster según el documento “Instalación y configuración del cluster de balanceo de carga para la capa de lógica de negocio de la aplicación Registro centralizado de donantes”

La prueba se ejecutó igual que la RCD_CP-CO-16 y permitió validar el funcionamiento correcto de la aplicación sobre la variante de la propuesta de solución diseñada para ella.

4.8. Conclusiones

Después de hacer realizado todas las pruebas necesarias para validar cada una de las variantes de la propuesta de solución se puede concluir que:

1. Las variantes de la propuesta de solución para las aplicaciones que utilizan el servidor de aplicación Apache + PHP se validaron completamente.

2. Las variantes de la propuesta de solución para las aplicaciones que utilizan el servidor de aplicación Apache-Tomcat solo fueron validadas parcialmente, debido a que no se le pudo realizar pruebas de carga.
3. Las pruebas realizadas mostraron que se cumplió el objetivo de la investigación y se contrastó la hipótesis.

CONCLUSIONES

A partir de la investigación realizada para obtener el diseño de una propuesta de solución que permita obtener tiempos de respuestas aceptables en los servidores web sobre los que se ejecutan las aplicaciones médicas cubanas, se concluyó que:

1. La implementación de la propuesta de solución basada en cluster de balanceo de carga para servidores web se realizaría utilizando la combinación del balanceador de carga IPVS con el Ldirectod y Heartbeat.

2. Entre las aplicaciones médicas de tipo web que se encuentran en desarrollo o han sido desarrolladas por la Facultad 7 de la UCI y/o Softel, se identificaron 25 que pueden necesitar esta solución y por sus características se dividieron en tres grupos.

3. El diseño de una propuesta de solución basada en cluster de balanceo de carga para servidores web, es específico para cada capa y/o nivel en que está dividida una aplicación web.

4. Las variantes de la propuesta de solución diseñadas no afectan el funcionamiento de las capas de las aplicaciones web que sobre ellas se ejecutan y cumplen con el objetivo de la investigación logrando tiempos de respuestas aceptables.

5. Las pruebas a los diseños de soluciones basadas en cluster de balanceo de carga son necesarias para su validación y ayudan a su perfeccionamiento.

RECOMENDACIONES

- Evaluar el desempeño de las aplicaciones web que ya están desplegadas para identificar si alguna requiere la o las variantes de la propuesta de solución diseñadas para ella.

- Lograr que los usuarios de la red del MINSAP, al navegar por las aplicaciones médicas cubanas de tipo web, accedan directamente a ellas y no a través de los servidores proxy de las instituciones, provincia o país.

- Estudiar las características del servidor de aplicación Apache-Tomcat para lograr que la información de estado de las sesiones de los usuarios se almacenen en los ficheros instantáneamente.

- Profundizar en la planificación del despliegue y gestión de cada una de las variantes de la propuesta de solución diseñadas y crear los procedimientos para realizar estas actividades.

REFERENCIAS BIBLIOGRÁFICAS

Apache Software Foundation. (2008). "Apache JMeter." Retrieved Marzo, 2008, from http://jakarta.apache.org/jmeter/.

Bourke, T. (2001). Seever Load Balancing. Sebastopol, O'Reilly & Associates, Inc.

Coda Project. (2008). "Coda File System." Retrieved Marzo, 2008, from http://www.coda.cs.cmu.edu/about.html.

Debian Project. (2008a). "Package: iptables (1.3.6.0debian1-5)." Retrieved Abril, 2008, from http://packages.debian.org/etch/iptables.

Debian Project. (2008b). "Package: rrdtool (1.2.15-0.3)." Retrieved Abril, 2008, from http://packages.debian.org/etch/rrdtool.

Debian Project. (2008c). "Package: snmp (5.2.3-7etch2)." Retrieved Abril, 2008, from http://packages.debian.org/etch/snmp.

Debian Project. (2008d). "Package: snmpd (5.2.3-7etch2)." Retrieved Abril, 2008, from http://packages.debian.org/etch/snmpd.

Godard, S. (2008). "SysStat Documentation." Abril, from http://pagesperso-orange.fr/sebastien.godard/documentation.html.

Horman, S. and J. Rief. (2006, January 10, 2008). "Ldirectord." Retrieved Febrero, 2008, from http://www.vergenet.net/linux/ldirectord/.

Kopper, K. (2005). Linux Enterprise Cluster. San Francisco, No Starch Press.

Kubat, K. (2008a). "Crossroads Documentation." Retrieved Febrero, 2008, from http://crossroads.e-tunity.com/servedoc.xr?format=html.

Kubat, K. (2008b). "Welcome to Crossroads." Retrieved Febrero, 2008, from http://crossroads.e-tunity.com/index.xr.

Linux HA Project. (2008). "Heartbeat." Retrieved Marzo, 2008, from http://www.linux-ha.org/Heartbeat.

Linux Virtual Server Project. (2005a). "Job Scheduling Algorithms in Linux Virtual Server." Retrieved Febrero, 2008, from http://www.linuxvirtualserver.org/docs/scheduling.html.

Linux Virtual Server Project. (2005b). "What is virtual server?" Retrieved Febrero, 2008, from http://www.linuxvirtualserver.org/whatis.html.

Linux Virtual Server Project. (2005c). "Why virtual server?" Retrieved Febrero, 2008, from http://www.linuxvirtualserver.org/why.html.

Linux Virtual Server Project. (2006). "KTCPVS." Retrieved Febrero, 2008, from http://www.linuxvirtualserver.org/software/ktcpvs/ktcpvs.html.

Mark, B., T. Sterling, et al. (2000). Cluster Computing White Paper, University of Portsmouth.

Martínez Jiménez, M. and G. Bergés Pujol. (2003). "Arquitecturas de Clustering de Alta Disponibilidad y Escalabilidad (Linux Virtual Server), ACADE (LVS)." Retrieved Marzo, 2008, from http://idefix.eup.uva.es/Manuales/Clustering/ACADE-LVS-memoria.pdf.

Meier, J. D., C. Farre, et al. (2007a). "How To: Stress Test Web Applications." Retrieved Abril, 2008, from http://www.codeplex.com/PerfTesting/Wiki/View.aspx?title=How%20To%3a%20Stress%20Test%20Web%20Applications&referringTitle=Stress%20Testing.

Meier, J. D., C. Farre, et al. (2007b). Performance Testing Guidance for Web Applications, Microsoft Corporation.

Nagios Enterprises. (2008). "About Nagios." Retrieved Abril, 2008, from http://www.nagios.org/about/.

Net-SNMP Project. (2007). "Net-SNMP." Retrieved Abril, 2008, from http://net-snmp.sourceforge.net/.

Oetiker, T. (2008). "What is MRTG ?" Retrieved Abril, 2008, from http://oss.oetiker.ch/mrtg/doc/mrtg.en.html.

Peña Montero, A. M., J. E. Coss Piña, et al. (2007). Cluster de servidores de aplicación para EVA UCI. Uciencia 2007, Ciudad de la Habana, UCI.

Pujol García, J. C., A. M. Peña Montero, et al. (2007). Clusters de servidores por balanceo de carga. Uciencia 2007, Ciudad de la Habana, UCI.

Rational Unified Process. (2003). "Concepts: Types of Test." Retrieved Abril, 2008, from http://rup.hops-fp6.org/process/workflow/test/co_tytst.htm.

Sánchez Rodríguez, A. and F. Pompa Sourd (2007). Arquitectura, normas y tecnologías para el desarrollo de aplicaciones informáticas para la Salud Pública en Cuba, MINSAP.

Sandberg, R., D. Goldberg, et al. (1985). Design and Implementation of the SUN Network Filesystem. USENIX Conference, Portland, USENIX.

Sloan, J. D. (2004). High Performance Linux Clusters with OSCAR, Rocks, OpenMosix, and MPI. Sebastopol, O'Reilly Media, Inc.

Softel (2006). Documento sobre la arquitectura de software para los componentes a emplear por el sistema de información para la salud, Softel.

Srisuresh, P. and M. Holdrege (1999). RFC 2663 - Terminología y consideraciones sobre Traducción de Direcciones IP, Network Working Group.

Tarreau, W. (2008). "HAProxy. The Reliable, High Performance TCP/HTTP Load Balancer." Retrieved Febrero, 2008, from http://haproxy.1wt.eu/.

BIBLIOGRAFÍA

1. Apache Software Foundation. (2005). "Jmeter User`s manual." from http://jakarta.apache.org/jmeter/index.html.
2. Apache Software Foundation. (2008). "Apache JMeter." from http://jakarta.apache.org/jmeter/.
3. Booth, A. and A. Citron. (2001). "Stress testing your software without stressing your testers." from http://www.ibm.com/developerworks/webservices/library/ibm-stress/.
4. Bourke, T. (2001). Seever Load Balancing. Sebastopol, O'Reilly & Associates, Inc.
5. Cisco Corporation. "Understanding load balancing algorithms." from http://www.cisco.com/application/pdf/paws/28580/lb_algorithms.pdf.
6. Cisco Corporation. "Understanding load balancing algorithms." from http://www.cisco.com/application/pdf/paws/28580/lb_algorithms.pdf.
7. Coda Project. (2008). "Coda File System." from http://www.coda.cs.cmu.edu/about.html.
8. Comer, D. E. (1996). Redes globales de información con Intenet y TCP-IP. Principios básicos, protocolos y arquitectura., Prentice Hall.
9. Debian Project. (2008). "Package: iptables (1.3.6.0debian1-5)." from http://packages.debian.org/etch/iptables.
10. Debian Project. (2008). "Package: rrdtool (1.2.15-0.3)." from http://packages.debian.org/etch/rrdtool.
11. Debian Project. (2008). "Package: snmp (5.2.3-7etch2)." from http://packages.debian.org/etch/snmp.
12. Debian Project. (2008). "Package: snmpd (5.2.3-7etch2)." from http://packages.debian.org/etch/snmpd.
13. Godard, S. (2008). "SysStat Documentation." from http://pagesperso-orange.fr/sebastien.godard/documentation.html.
14. Horman, S. and J. Rief. (2006, January 10, 2008). "Ldirectord." from http://www.vergenet.net/linux/ldirectord/.
15. Kopper, K. (2005). Chapter 6. Heartbeat Introduction and theory. The Linux Enterprise Cluster. W. Pollock, No Starch Press.

16. Kopper, K. (2005). Linux Enterprise Cluster. San Francisco, No Starch Press.

17. Kubat, K. (2008). "Crossroads Documentation." from http://crossroads.e-tunity.com/servedoc.xr?format=html.

18. Kubat, K. (2008). "Welcome to Crossroads." from http://crossroads.e-tunity.com/index.xr.

19. Leinwand, A. and K. F. Conroy (1996). Network management - A practical perspective, Cisco Sistems.

20. Linux HA Project. (2008). "Heartbeat." from http://www.linux-ha.org/Heartbeat.

21. Linux Virtual Server Project. (2005). "Job Scheduling Algorithms in Linux Virtual Server." from http://www.linuxvirtualserver.org/docs/scheduling.html.

22. Linux Virtual Server Project. (2005). "What is virtual server?" from http://www.linuxvirtualserver.org/whatis.html.

23. Linux Virtual Server Project. (2005). "Why virtual server?" from http://www.linuxvirtualserver.org/why.html.

24. Linux Virtual Server Project. (2006). "KTCPVS." from http://www.linuxvirtualserver.org/software/ktcpvs/ktcpvs.html.

25. Mack, J. (2007). "LVS-HOWTO." from http://www.austintek.com/LVS/LVS-HOWTO/HOWTO/.

26. Mack, J. (2007). "LVS-mini-HOWTO." from http://www.austintek.com/LVS/LVS-HOWTO/mini-HOWTO/LVS-mini-HOWTO.html.

27. Mark, B., T. Sterling, et al. (2000). Cluster Computing White Paper, University of Portsmouth.

28. Martínez Jiménez, M. and G. Bergés Pujol. (2003). "Arquitecturas de Clustering de Alta Disponibilidad y Escalabilidad (Linux Virtual Server), ACADE (LVS)." from http://idefix.eup.uva.es/Manuales/Clustering/ACADE-LVS-memoria.pdf.

29. Meier, J. D., C. Farre, et al. (2007). "How To: Stress Test Web Applications." from http://www.codeplex.com/PerfTesting/Wiki/View.aspx?title=How%20To%3a%20Stress%20Test%20Web%20Applications&referringTitle=Stress%20Testing.

30. Meier, J. D., C. Farre, et al. (2007). Performance Testing Guidance for Web Applications, Microsoft Corporation.

31. Myers, G. J., T. Badgett, et al. (2004). The art of software testing. New Jersey, USA., John Willey & Sons Inc.

32. Nagios Enterprises. (2008). "About Nagios." from http://www.nagios.org/about/.

33. Net-SNMP Project. (2007). "Net-SNMP." from http://net-snmp.sourceforge.net/.

34. Oetiker, T. (2008). "What is MRTG ?" from http://oss.oetiker.ch/mrtg/doc/mrtg.en.html.

35. Pawul, R. (2003). "Getting Started with Linux-HA (Heartbeat)." from http://www.linux-ha.org/GettingStarted.

36. Peña Montero, A. M., J. E. Coss Piña, et al. (2007). Cluster de servidores de aplicación para EVA UCI. Uciencia 2007, Ciudad de la Habana, UCI.

37. Ponnachath, P. (2002). "Application server benchmarking fundamentals." from http://www.sybase.com/content/1018914/Benchmarking_fundamentals_v3.pdf.

38. Pujol García, J. C., A. M. Peña Montero, et al. (2007). Clusters de servidores por balanceo de carga. Uciencia 2007, Ciudad de la Habana, UCI.

39. Rational Unified Process. (2003). "Concepts: Types of Test." from http://rup.hops-fp6.org/process/workflow/test/co_tytst.htm.

40. Sánchez Rodríguez, A. and F. Pompa Sourd (2007). Arquitectura, normas y tecnologías para el desarrollo de aplicaciones informáticas para la Salud Pública en Cuba, MINSAP.

41. Sandberg, R., D. Goldberg, et al. (1985). Design and Implementation of the SUN Network Filesystem. USENIX Conference, Portland, USENIX.

42. Sloan, J. D. (2004). High Performance Linux Clusters with OSCAR, Rocks, OpenMosix, and MPI. Sebastopol, O'Reilly Media, Inc.

43. Softel (2006). Documento sobre la arquitectura de software para los componentes a emplear por el sistema de información para la salud, Softel.

44. Srisuresh, P. and M. Holdrege (1999). RFC 2663 - Terminología y consideraciones sobre Traducción de Direcciones IP, Network Working Group.

45. Stalling, W. Comunicaciones y redes de computadores., Prentice Hall.

46. Tanenbaum, A. S. (2003). Computer Networks., Prentice Hall.

47. Tarreau, W. (2008). "HAProxy. The Reliable, High Performance TCP/HTTP Load Balancer." from http://haproxy.1wt.eu/.

ANEXO 1. Documentos de diseño de la propuesta de solución

En este anexo se listan los documentos de diseño de la propuesta de solución. Estos documentos se adjuntan a este informe en formato digital y se entregan en un CD-ROM.

Instalación y configuración de un cluster de balanceo de carga genérico para servidores web.

Nombre del archivo: Solución genérica.pdf
Ubicación: Soluciones\Solución genérica\
Ubicación de los ficheros de configuración adjuntos: Soluciones\Solución genérica\ficheros de configuración\

Cluster de balanceo de carga para la capa de presentación de la aplicación Registro de personal de la salud.

Nombre del archivo: Solución RPS-CP.pdf
Ubicación: Soluciones\Soluciones para RPS\Capa de presentación\
Ubicación de los ficheros de configuración adjuntos: Soluciones\Soluciones para RPS\Capa de presentación\ficheros de configuración\

Cluster de balanceo de carga para la capa de lógica de negocio de la aplicación Registro de personal de la salud.

Nombre del archivo: Solución RPS-LN.pdf
Ubicación: Soluciones\Soluciones para RPS\Capa de lógica de negocio\
Ubicación de los ficheros de configuración adjuntos: Soluciones\Soluciones para RPS\Capa de lógica de negocio\ficheros de configuración\

Cluster de balanceo de carga para la aplicación Balance Material.

Nombre del archivo: Solución BM.pdf
Ubicación: Soluciones\Solución para Balance Material\
Ubicación de los ficheros de configuración adjuntos: Soluciones\Solución para Balance Material\ficheros de configuracion

Cluster de balanceo de carga para la capa de presentación de la aplicación Registro centralizado de donantes.

Nombre del archivo: Solución RCD-CP.pdf
Ubicación: Soluciones\Soluciones para RCD\Capa de presentación\
Ubicación de los ficheros de configuración adjuntos: Soluciones\Soluciones para RCD\Capa de presentación\ficheros de configuración\

Cluster de balanceo de carga para la capa de lógica de negocio de la aplicación Registro centralizado de donantes.

Nombre del archivo: Solución RCD-LN.pdf
Ubicación: Soluciones\Soluciones para RCD\Capa de lógica de negocio\
Ubicación de los ficheros de configuración adjuntos: Soluciones\Soluciones para RCD\Capa de lógica de negocio\ficheros de configuración\

ANEXO 2. Esquema de codificación en las referencias a las pruebas

Para crear el esquema de codificación utilizado en las referencias de las pruebas se tuvo en cuenta la siguiente sintaxis:

siglas del nombre de la aplicación[_capa de la aplicación]-tipo de prueba-número de la prueba

El campo “siglas del nombre de la aplicación” indica las siglas con las que se conoce la aplicación que está siendo probada. Este campo puede tener las cadenas: GEN, para un diseño genérico, RPS para la aplicación Registro de personal de la salud, BM para la aplicación Balance Material y RCD para la aplicación Registro centralizado de donantes.

El campo opcional “capa de la aplicación” se utiliza para especificar, cuando las pruebas no se le realizan a todas las capas posibles de la aplicación, cual es la que se esta sometiendo a prueba. Este campo puede tener las cadenas: CP para indicar la capa de presentación y LN para indicar la capa de lógica de negocio.

El campo “tipo de prueba” se utiliza para indicar el tipo de la prueba que se está realizando. Puede tener las cadenas: CO para indicar pruebas de configuración y CA para indicar pruebas de carga.

El campo “número de la prueba” indica el número de la prueba entre todas las realizadas.

ANEXO 3. Documentos de planificación y análisis de resultado de las pruebas

En este anexo se listan los documentos de planificación y análisis de resultado de las pruebas realizadas durante la investigación. Estos documentos se adjuntan a este informe en formato digital y se entregan en un CD-ROM.

Prueba GEN-CO-01

Nombre del archivo de planificación de la prueba: GEN-CO-01.pdf
Nombre del archivo de análisis de resultados de la prueba: GEN-CO-01_AR.pdf
Ubicación de los archivos: Pruebas\Pruebas a Diseño genérico\

Prueba GEN-CO-02

Nombre del archivo de planificación de la prueba: GEN-CO-02.pdf
Nombre del archivo de análisis de resultados de la prueba: GEN-CO-02_AR.pdf
Ubicación de los archivos: Pruebas\Pruebas a Diseño genérico\

Prueba GEN-CO-03

Nombre del archivo de planificación de la prueba: GEN-CO-03.pdf
Nombre del archivo de análisis de resultados de la prueba: GEN-CO-03_AR.pdf
Ubicación de los archivos: Pruebas\Pruebas a Diseño genérico\

Prueba RPS_CP-CO-04

Nombre del archivo de planificación de la prueba: RPS_CP-CO-04.pdf
Nombre del archivo de análisis de resultados de la prueba: RPS_CP-CO-04_AR.pdf
Ubicación de los archivos: Pruebas\Pruebas a RPS\Capa de presentación\

Prueba RPS_CP-CO-05

Nombre del archivo de planificación de la prueba: RPS_CP-CO-05.pdf
Nombre del archivo de análisis de resultados de la prueba: RPS_CP-CO-05_AR.pdf
Ubicación de los archivos: Pruebas\Pruebas a RPS\Capa de presentación\

Prueba RPS_CP-CA-06

Nombre del archivo de planificación de la prueba: RPS_CP-CA-06.pdf

Nombre del archivo de análisis de resultados de la prueba: RPS_CP-CA-06_AR.pdf
Ubicación de los archivos: Pruebas\Pruebas a RPS\Capa de presentación\

Prueba RPS_CP-CA-07

Nombre del archivo de planificación de la prueba: RPS_CP-CA-07.pdf
Nombre del archivo de análisis de resultados de la prueba: RPS_CP-CA-07_AR.pdf
Ubicación de los archivos: Pruebas\Pruebas a RPS\Capa de presentación\

Prueba RPS_LN-CO-08

Nombre del archivo de planificación de la prueba: RPS_LN-CO-08.pdf
Nombre del archivo de análisis de resultados de la prueba: RPS_LN-CO-08_AR.pdf
Ubicación de los archivos: Pruebas\Pruebas a RPS\Capa de lógica de negocio\

Prueba RPS_LN-CO-09

Nombre del archivo de planificación de la prueba: RPS_LN-CO-09.pdf
Nombre del archivo de análisis de resultados de la prueba: RPS_LN-CO-09_AR.pdf
Ubicación de los archivos: Pruebas\Pruebas a RPS\Capa de lógica de negocio\

Prueba RPS_LN-CA-10

Nombre del archivo de planificación de la prueba: RPS_LN-CA-10.pdf
Nombre del archivo de análisis de resultados de la prueba: RPS_LN-CA-10_AR.pdf
Ubicación de los archivos: Pruebas\Pruebas a RPS\Capa de lógica de negocio\

Prueba RPS_LN-CA-11

Nombre del archivo de planificación de la prueba: RPS_LN-CA-11.pdf
Nombre del archivo de análisis de resultados de la prueba: RPS_LN-CA-11_AR.pdf
Ubicación de los archivos: Pruebas\Pruebas a RPS\Capa de lógica de negocio\

Prueba BM-CO-12

Nombre del archivo de planificación de la prueba: BM-CO-12.pdf
Nombre del archivo de análisis de resultados de la prueba: BM-CO-12_AR.pdf
Ubicación de los archivos: Pruebas\Pruebas a Balance Material\

Prueba BM-CO-13

Nombre del archivo de planificación de la prueba: BM-CO-13.pdf

Nombre del archivo de análisis de resultados de la prueba: BM-CO-13_AR.pdf

Ubicación de los archivos: Pruebas\Pruebas a Balance Material\

Prueba BM-CA-14

Nombre del archivo de planificación de la prueba: BM-CA-14.pdf

Nombre del archivo de análisis de resultados de la prueba: BM-CA-14_AR.pdf

Ubicación de los archivos: Pruebas\Pruebas a Balance Material\

Prueba BM-CA-15

Nombre del archivo de planificación de la prueba: BM-CA-15.pdf

Nombre del archivo de análisis de resultados de la prueba: BM-CA-15_AR.pdf

Ubicación de los archivos: Pruebas\Pruebas a Balance Material\

Prueba RCD_CP-CO-16

Nombre del archivo de planificación de la prueba: RCD_CP-CO-16.pdf

Nombre del archivo de análisis de resultados de la prueba: RCD_CP-CO-16_AR.pdf

Ubicación de los archivos: Pruebas\Pruebas a RCD\Capa de presentación\

Prueba RCD_LN-CO-17

Nombre del archivo de planificación de la prueba: RCD_LN-CO-17.pdf

Nombre del archivo de análisis de resultados de la prueba: RCD_LN-CO-17_AR.pdf

Ubicación de los archivos: Pruebas\Pruebas a RCD\Capa de lógica de negocio\