Universidad de las Ciencias Informáticas

Facultad 7

Título: Cluster de servidores de bases de datos para aplicaciones web, sobre software libre

Trabajo de Diploma para optar por el título de
Ingeniero en Ciencias Informáticas

Autores: Dayrel Almaguer Chávez

Dysán García Riera

Tutor: Ing. Juan Carlos Pujol García

Ciudad de La Habana, Julio del 2008

"Año 50 de la Revolución"

DECLARACIÓN DE AUTORÍA

Declaramos ser los autores del presente trabajo y reconocemos a la Facultad 7 de la Universidad de las Ciencias Informáticas y a la empresa Softel ha hacer uso del mismo en su beneficio.

Y nos comprometemos a no utilizar el mismo con otros fines que no sean los autorizados por las instituciones antes mencionadas.

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

Autores:

_____ _____

Dayrel Almaguer Chávez Dysán García Riera

Tutor:

_____

Ing. 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

AGRADECIMIENTOS

Dayrel Almaguer Chávez:

A mis padres, por haberme brindado todo su apoyo en la vida y por la confianza que siempre han depositado en mí.

A mi hermano, porque su ejemplo me ha servido de guía en mi formación como profesional.

A mis amigos, por estar presentes en cada momento y por brindarme su ayuda incondicional.

Al tutor de este trabajo, Juan Carlos, por haber contribuido a su realización.

A mi compañero de tesis Dysán, por el esfuerzo y la ayuda que me brindó.

A Adrián, porque también colaboró en la realización de este trabajo.

A los profesores de la Universidad de las Ciencias Informáticas (UCI), por todo lo que me han enseñado.

A todos los que han estado a mi lado brindándome su apoyo durante estos años.

Dysán García Riera:

A mi madre y a mis hermanos, por la confianza que han depositado en mí.

A mis amigos, por haber estado presentes en todo momento.

Al tutor de este trabajo, Juan Carlos, por haber contribuido a su realización.

A Adrián, porque su ayuda ha sido muy valiosa en la realización de este trabajo.

A mi compañero de tesis Dayrel, por el esfuerzo, la ayuda que me brindó y sobre todo por ser un gran amigo.

A todos los profesores y trabajadores de la universidad que de una forma u otra han influido en mi formación como profesional.

DEDICATORIA

Dayrel Almaguer Chávez:

A mis padres.

A mi hermano.

A mis amigos de siempre.

Dysán García Riera:

A la memoria de mi padre.

Muy en especial a mi madre.

A toda mi familia.

A mis amigos de la universidad.

RESUMEN

Esta investigación tiene como objetivo la búsqueda de soluciones de cluster de bases de datos para las aplicaciones médicas del Ministerio de Salud Pública (MINSAP) cuyo despliegue se realiza de forma centralizada. Surge por la necesidad de elevar el rendimiento de los servidores de bases de datos de dichas aplicaciones.

Actualmente en Cuba no se desarrolla ninguna tecnología de este tipo por lo que se realizó un estudio de las soluciones de software libre disponibles para confeccionar un cluster de servidores de bases de datos. Las principales herramientas utilizadas para confeccionar las soluciones fueron los *middlewares* Myosotis y Sequoia así como las técnicas de balanceo de carga de LVS y la replicación de datos de MySQL. Se diseñaron cuatro soluciones de cluster para las aplicaciones médicas, desarrolladas por la Universidad de las Ciencias Informáticas (UCI) y la empresa Softel, teniendo en cuenta las características de cada una de ellas. Los diseños confeccionados fueron validados a través de un grupo de pruebas de configuración y de rendimiento.

Las soluciones de cluster propuestas contribuyeron a disminuir los tiempos de respuesta de las peticiones realizadas en un grupo importante de aplicaciones y disminuyeron el consumo de recursos en los servidores de bases de datos utilizados por las mismas. Durante la ejecución de las pruebas se detectaron ineficiencias en el código de implementación de las aplicaciones y violaciones en el cumplimiento de sus requisitos no funcionales.

PALABRAS CLAVE

Cluster, aplicaciones, pruebas, rendimiento, tiempo de respuesta, balanceo de carga, servidores de bases de datos.

TABLA DE CONTENIDOS

INTRODUCCIÓN

Desde el triunfo de la revolución, Cuba ha seguido un camino destinado a incorporar la informática y las nuevas tecnologías de la información y las comunicaciones (TIC) a la sociedad. Como parte de esta estrategia se ha desarrollado todo un proceso de informatización que tiene como objetivo elevar la eficiencia y calidad de los servicios que se brindan en varios sectores de la sociedad cubana.

La Salud Pública ha sido uno de los sectores priorizados en este proceso. Para lograr el cumplimiento de las metas propuestas por el gobierno cubano un grupo de instituciones trabajan en elevar el desarrollo y calidad del *software* en esta área, tal es le caso de la Universidad de las Ciencias Informáticas (UCI) y la empresa productora de *software* Softel.

Tanto los estudiantes y profesores de la universidad, como los especialistas de Softel han colaborado entre sí para desarrollar un grupo de aplicaciones en busca de la automatización de los procesos que se llevan a cabo a diario en las instituciones médicas. Uno de los objetivos que se persigue con la implementación de estas aplicaciones es el de unificar e integrar los datos generados en los distintos niveles de salud donde puede ser atendido un paciente. Por tal motivo se han desarrollado un grupo de aplicaciones de tipo web utilizando diferentes tecnologías que facilitan su despliegue de manera centralizada aprovechando las facilidades que brinda la red telemática del Sistema Nacional de Salud (SNS), Infomed.

Las aplicaciones manejan un gran volumen de información la cual se encuentra almacenada en un grupo de servidores de bases de datos. Estos servidores deben recibir constantemente solicitudes de servicio de las instituciones médicas localizadas en varias regiones geográficas distribuidas a lo largo de todo el país.

El aumento del número de usuarios que acceden a los servidores de bases de datos que almacenan la información produce un incremento de las solicitudes realizadas a dichos servidores. Esto provoca que la carga de trabajo de los mismos sea mayor por lo que los recursos que manejan deben ser suficientes para soportar dicha carga. Sin embargo el constante crecimiento del número de usuarios que explotan el sistema puede provocar que en algún momento se requiera de la utilización de más recursos de procesamiento que los que se encuentran disponibles en los servidores existentes.

Una posible solución en este caso sería mejorar algunas de sus características, tales como memoria RAM (*Random Acces Memory* / Memoria de Acceso Aleatorio), capacidad de almacenamiento en disco duro, frecuencia de CPU (*Central Processing Unit* / Unidad de Procesamiento Central), velocidad de las tarjetas de red o NICs (*Network Interface Card* / Tarjeta de Interfaz de Red), etc, pero esta solución estaría afectada por la relación costo - utilidad de dicho servidor.

Una posible solución, para aumentar el rendimiento en los servidores de bases de datos de las aplicaciones web centralizadas con que cuenta el Ministerio de Salud Pública (MINSAP) en Cuba, es la implementación de un cluster para la capa de datos de dichas aplicaciones. La búsqueda de este tipo de soluciones para las aplicaciones web desarrolladas para el sector de la salud es el tema principal en el cual se basa esta investigación.

Implementar soluciones de cluster proporcionaría grandes ventajas en cuanto a: recursos necesarios en los servidores y por tanto en su costo de adquisición, además se lograrían ofertar los servicios de manera más eficiente. La búsqueda de soluciones de este tipo se hará realizando un estudio de las tendencias actuales relacionadas con el tema y basado en los resultados del mismo se diseñarán y se probarán varias soluciones.

La *situación problémica* que se tiene es la siguiente**:** El proceso de informatización que se lleva a cabo en el sector de la salud se hace extensible cada día, como parte de dicho proceso se han desplegado un grupo de aplicaciones web a nivel central y se trabaja en la implementación de otras aplicaciones con el mismo propósito. Producto a ello se estima que existirá un incremento del número de usuarios que hacen uso de las aplicaciones y la demanda de servicios que se ofertan será también mayor, esto va a provocar que los servidores de bases de datos, que contienen toda la información, sufran en muchas ocasiones sobrecargas de trabajo producto a la cantidad y complejidad de solicitudes que deben atender de manera concurrente.

Cuando esto ocurra se producirá un deterioro en los tiempos de respuesta de las peticiones realizadas por los usuarios y en el peor de los casos pudiera producirse el colapso de alguno de los servidores, dejando así de ofrecer los servicios que prestan y afectando directamente a los usuarios del sistema.

El *problema científico* que se plantea radica en que los servidores de bases de datos de las aplicaciones médicas, desplegados a nivel central, tienen un bajo rendimiento en comparación con la carga de trabajo que deben soportar.

Para dar solución al problema planteado se define como *objeto de estudio* los servidores de bases de datos de las aplicaciones web centralizadas, ya desplegadas o en desarrollo, con que cuenta el Ministerio de Salud Pública (MINSAP) y se centra la investigación en el rendimiento de dichos servidores, lo cual se enmarca como *campo de acción*.

El *objetivo general* de esta investigación es elaborar propuestas de soluciones de software libre para aumentar el rendimiento de los servidores de bases de datos de las aplicaciones web centralizadas del MINSAP.

Se plantea la siguiente *idea a defender*: La utilización de una tecnología de cluster de bases de datos en los servidores de las aplicaciones médicas centralizadas del MINSAP aumentará, en la mayoría de los casos, el rendimiento de dichos servidores sin afectar las funcionalidades de las aplicaciones.

Para cumplir con el objetivo propuesto se han trazado un grupo de *tareas de la investigación* entre las cuales se encuentran:

1. Analizar las variantes de clusters de servidores de bases de datos más utilizadas en el mundo del software libre.
2. Clasificar el conjunto de aplicaciones médicas en diferentes tipos, según características comunes.
3. Seleccionar herramientas de prueba para medir stress en las soluciones de cluster de bases de datos propuestas.
4. Diseñar solución o conjunto de soluciones de cluster de bases de datos para cada tipo de aplicación.
5. Desplegar un escenario de pruebas en un laboratorio con copias de las aplicaciones reales con sus respectivas bases de datos.
6. Configurar la solución de cluster de base de datos, en un laboratorio, según la aplicación que esté a prueba.
7. Realizar pruebas de configuración, en un laboratorio, a las soluciones de cluster de bases de datos desarrolladas para cada tipo de aplicación.
8. Realizar pruebas de rendimiento, en un laboratorio, a las soluciones de cluster de bases de datos desarrolladas para cada tipo de aplicación.
9. Implementar alta disponibilidad y escalabilidad a las soluciones desarrolladas, de ser posible.
10. Proponer al menos una solución de cluster para cada tipo de aplicación.

El presente trabajo de diploma está estructurado en cuatro capítulos, cuyo contenido se presenta a continuación:

En el capítulo uno se explican conceptos generales relacionados con la tecnología de cluster y se describen herramientas de software libre utilizadas para implementar clusters de servidores de bases de datos así como para ofrecer alta disponibilidad y realizar tareas de monitoreo. Se define además un grupo de pruebas para medir el rendimiento y las funcionalidades de los clusters.

En el capítulo dos se describe de forma breve las características de las aplicaciones desarrolladas por la Universidad de las Ciencias Informáticas (UCI) y Softel para las cuales se proponen las soluciones de cluster de bases de datos abordadas en esta investigación.

El capítulo tres se refiere de manera detallada al diseño de las soluciones de cluster de bases de datos propuestas para cada grupo de aplicaciones. Se presenta además una propuesta de despliegue y un mecanismo de monitoreo y gestión de la seguridad para cada solución.

En el capítulo cuatro se describen las pruebas necesarias para validar las soluciones de cluster. Se define un procedimiento de prueba general así como las métricas a medir y la planificación de cada prueba. Además se realiza un análisis de los resultados obtenidos durante la validación de cada propuesta de solución.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

1.1. Introducción a la tecnología de cluster

Actualmente el manejo de información que se realiza a través de los equipos de cómputo crece de manera extraordinaria convirtiéndose en tareas complejas que requieren de alto procesamiento computacional, dicho procesamiento no se logra en muchos casos utilizando un simple ordenador de escritorio, para ello se requiere un sistema con características mejoradas de *hardware* cuya capacidad le posibilite aumentar su desempeño en el manejo y procesamiento de flujos muy altos de información, estos sistemas son llamados “supercomputadoras”.

Si el número de usuarios de una aplicación web es muy elevado se necesitan máquinas capaces de procesar todas las peticiones realizadas de manera concurrente y que den respuesta a dichas peticiones en tiempos aceptables para el usuario final. El uso de un sistema como el descrito anteriormente está limitado por los recursos de *hardware* que posee, esto puede ser una fuerte limitante cuando la concurrencia de usuarios que navegan en la aplicación aumenta con el paso del tiempo, ya que dichos recursos pueden llegar a ser insuficientes, la solución sería entonces agregar nuevos componentes de *hardware* para aumentar las capacidades de procesamiento o adquirir una máquina con mejores características. Sin embargo en ambos casos los costos de adquisición son altos y se suman a estos los gastos en que se incurre para el mantenimiento sistemático de *software* y de *hardware*.

Gracias a los avances de la informática y las tecnologías, comprar una máquina potente a precios elevados no es la única solución. Existen otras formas de mejorar el rendimiento de los servidores de las aplicaciones web, entre ellas se encuentra la implementación de un cluster de servidores, esta constituye una solución económicamente más rentable que el uso de una supercomputadora y brinda las mismas funcionalidades con una mayor flexibilidad.

1.2. Características generales de la tecnología de cluster

Definición

Un cluster es un conjunto de computadoras, a menudo con semejantes componentes de *hardware*, que se interconectan entre sí a través de un sistema de red de alta velocidad y son capaces de elevar la eficiencia para realizar determinadas tareas que individualmente no podrían realizar debido a la creciente necesidad de potencia computacional que demandan algunas aplicaciones. A pesar de estar compuesto por varias computadoras, la arquitectura de un cluster es completamente transparente para

el usuario final, por lo que este lo ve como un único sistema que representa una máquina con altos niveles de procesamiento.

Ventajas que ofrece

La implementación de un cluster brinda gran flexibilidad dada por las características de sus nodos. Los nodos de un cluster pueden tener las mismas configuraciones de *hardware* y sistema operativo, cuando esto ocurre se le llama cluster homogéneo. También puede que cada uno de ellos alcance un rendimiento diferente, proporcionado por la variedad de características de *hardware*, pero con arquitecturas y sistemas operativos similares, en cuyo caso el cluster se denomina cluster semi-homogéneo. Como última alternativa cuando los nodos tienen diferente *hardware* y sistema operativo se le llama cluster heterogéneo. La flexibilidad de un cluster hace más fácil y económica su construcción.

Cada uno de los nodos de un cluster puede ser un sistema completo para usar un amplio rango de aplicaciones, por ejemplo, un nodo puede constituir un servidor de bases de datos y contener la información utilizada por diferentes aplicaciones, estos nodos se pueden reemplazar fácilmente en caso de que su funcionamiento no sea correcto e incluso se pueden integrar nuevos nodos a la granja de servidores lo que convierte al cluster en un sistema altamente escalable. Esta característica permite al sistema amoldarse en todo momento según las necesidades existentes, que pueden ser cambiantes, y hacerlo con el mínimo costo.

Además se pueden formar sistemas verdaderamente grandes que comprenden desde dos hasta varios cientos de nodos lo que aumenta la disponibilidad de los servicios y hace que estos se brinden de forma más eficiente. El escalamiento de un cluster puede producirse de manera vertical u horizontal. Generalmente el escalar verticalmente o escalar hacia arriba, significa el añadir más recursos a un nodo en particular mientras que el escalado horizontal significa agregar más nodos al sistema.

Hoy en día las llamadas supercomputadoras son equipos excesivamente caros mientras que la implementación de un cluster resulta más barata y económica. Se puede confeccionar un cluster utilizando simples ordenadores de escritorio que llegue a ofrecer rendimiento muy cercano al alcanzado por una supercomputadora en cuanto a poder de cómputo. Por otra parte el *hardware* de red necesario para la interconexión de los nodos del cluster experimenta a medida que transcurre el tiempo un decremento constante de precio incluso se pueden lograr ahorros adicionales empleando un solo monitor, *mouse* y teclado para la administración de todo el sistema.

Clasificación de los clusters

De acuerdo a los servicios que presta un cluster puede ser clasificado de tres formas:

- Cluster de alta disponibilidad.
- Cluster de alto rendimiento.
- Cluster de balanceo de carga.

Un *cluster de alta disponibilidad* tiene el objetivo de proporcionar máxima disponibilidad y confiabilidad de los servicios que ofrece de tal manera que estos se brinden ininterrumpidamente. Se consigue alta disponibilidad haciendo redundantes los sistemas y de esta forma cuando se produce la caída del nodo que brinda el servicio un clon del mismo puede tomar el control y comenzar a servir el servicio nuevamente. La confiabilidad se consigue mediante un *software* que detecta fallos y permite la recuperación frente a los mismos. Un cluster de alta disponibilidad evita que el sistema tenga un único punto vulnerable a la ocurrencia de fallos.

En un *cluster de alto rendimiento* se ejecutan tareas que requieren de gran capacidad computacional, grandes cantidades de memoria, o ambos a la vez. Este tipo de cluster divide las tareas en tareas más pequeñas y las reparte entre los nodos que lo conforman para que sean calculadas en ellos y así agilizar el procesamiento de los datos. Estos sistemas se implementan en un ambiente de programación paralela utilizando algoritmos que hacen uso de recursos compartidos tales como CPU, memoria, datos y servicios. Entre las aplicaciones más comunes de clusters de alto rendimiento se encuentra el pronóstico numérico del estado del tiempo, astronomía, investigación en criptografía, análisis de imágenes, entre otras.

El *cluster de balanceo de carga* se encarga de colocar en paralelo varios servidores (servidores reales) capaces de brindar el mismo servicio y de alguna forma repartir el trabajo entre ellos, tal que los clientes vean servidas sus solicitudes en tiempos menores y aceptables. Los clientes deben ver al conjunto de servidores como si fuera uno solo (servidor virtual). Un cluster de balanceo de carga tiene peculiaridades de cluster de alta disponibilidad y alto rendimiento. (Bourke 2001)

Empleo de técnicas de clustering para bases de datos en el mundo

En el mundo existen varios ejemplos del empleo de tecnologías de cluster para servidores de bases de datos. El ejemplo más significativo es el cluster de Google. Actualmente Google es el motor de búsqueda de Internet más grande y más usado; implementa un cluster de servidores compuesto por mas de 60 000 servidores repartidos en varios centros de datos los cuales se localizan en diversos puntos del planeta (Norteamérica, Asia y Europa). Cada uno de los servidores de datos de Google

tiene instalado un sistema operativo Linux y sobre ellos se realizan técnicas de balanceo de carga y replicación. El uso de un sistema distribuido de almacenamiento de datos y no centralizado garantiza un menor costo en la implementación del cluster y reduce las posibilidades de ocurrencia de fallos, por otra parte aumenta la escalabilidad y el rendimiento del sistema lo cual hace posible que puedan ser atendidas hasta 40 millones de búsquedas por día. (Barroso et al. 2003)

En Cuba no se conoce del uso de una tecnología de cluster para servidores de de bases de datos, aunque se emplean técnicas de *clustering* con otros fines. Tal es el caso del Centro de Ingeniería Genética y Biotecnología el cual cuenta con un cluster conformado por 128 procesadores. Este sistema está dedicado a numerosas tareas dentro de la investigación científica de la institución y para dar apoyo a otras investigaciones que requieren cálculos sobre ambientes de computación paralela. (Gutiérrez Díaz et al. 2003)

1.3. Tecnologías para construir un cluster de servidores de bases de datos

Un cluster puede ser aplicable en cualquiera de los niveles o capas en que se despliega una aplicación web, ya sea presentación, lógica de negocio o acceso a datos, incluso puede aplicarse en las tres al mismo tiempo si fuese necesario.

Existen varias herramientas de software libre para conformar un cluster de servidores en la capa de datos de las aplicaciones. A continuación se describen algunas de ellas.

1.3.1. SQL Relay

(Mouse 2006)

SQL Relay es una herramienta utilizada en sistemas operativos Unix y Linux que permite las conexiones de varios clientes a diferentes bases de datos, soportadas o no, para estas plataformas. Se puede establecer una conexión con las siguientes bases de datos: Oracle, MySQL, PostgreSQL, Sybase, MS SQL Server, IBM DB2, entre otras. Esta herramienta brinda APIs para varios lenguajes de programación como C, C++, Perl, Python, PHP, Ruby, Java, etc y librerías de reemplazo para clientes MySQL y PostgreSQL que permiten a las aplicaciones clientes de MySQL y PostgreSQL ejecutar consultas sobre bases de datos de otros proveedores como Oracle o MS SQL Server.

SQL Relay establece un *pool* de conexiones persistentes entre las aplicaciones y las bases de datos e implementa técnicas de balanceo de carga que permiten dirigir más conexiones a las máquinas con mayor nivel de procesamiento dentro del cluster, esta característica puede mejorar considerablemente el rendimiento de las bases de datos utilizadas por las aplicaciones web y posibilita que el cluster

pueda ser conformado por una mezcla heterogénea de computadoras. Otra facilidad que brinda es la migración de manera sencilla entre bases de datos de diferentes proveedores.

Su funcionamiento se basa en la ejecución de demonios. Los demonios de SQL Relay mantienen sesiones abiertas con las bases de datos, estos a su vez se relacionen con un demonio oyente que identifica las conexiones de los clientes. Cuando un cliente es identificado por un demonio oyente, este verifica si existe alguna conexión establecida y de ser así permite que el cliente pueda conectarse a través de ella haciendo uso de la sesión establecida por el demonio de conexión, de lo contrario el cliente debe esperar en una cola hasta que exista una conexión disponible.

Las bases de datos que conforman el cluster deben estar sincronizadas haciendo uso de alguna técnica de replicación independiente de SQL Relay.

Limitaciones de SQL Relay

Durante los experimentos realizados en el laboratorio para validar la funcionalidad de esta herramienta se pudo comprobar que SQL Relay define un grupo de funciones específicas que deben ser llamadas desde el código de implementación de la aplicación para poder acceder a las bases de datos del cluster. Esto quiere decir que una aplicación que haya sido implementada sin que previamente se definiera el uso de las funciones específicas de SQL Relay no podrá ser desplegada sobre el cluster si no se realizan modificaciones en el código de implementación de la misma.

También las pruebas realizadas en el laboratorio, mostraron inestabilidad y no cumplimiento de todo lo especificado en la documentación de esta herramienta.

1.3.2. MySQL Proxy

MySQL Proxy es un programa sencillo que se sitúa entre un cliente y un servidor MySQL para monitorizar, analizar o transformar sus comunicaciones. Entre sus funcionalidades más comunes se encuentra el balanceo de carga, análisis de fallas, análisis de consultas a las bases de datos, filtraje y modificación de consultas. En lugar de conectarse directamente al servidor MySQL, el cliente se conecta a esta herramienta que actúa como *proxy* y esta envía las consultas a los diferentes nodos del cluster.

Haciendo una analogía, para entender mejor el funcionamiento de MySQL Proxy, se puede decir que este recibe un cubo vacío del cliente (una consulta), busca los datos en el servidor y llena el cubo con ellos, luego lo transmite de vuelta al cliente. (Maxia 2007)

Su funcionamiento consiste en monitorear y modificar las comunicaciones entre el cliente y los servidores que conforman el cluster, para ello intercepta las consultas utilizando el lenguaje de

scripting LUA y define qué hacer con ellas antes de que se envíen hacia el servidor o sean devueltas directamente al cliente. De esta manera el *proxy* puede enviar consultas adicionales a los servidores y el cliente final obtiene la respuesta de la consulta original y además algunos datos informativos, como por ejemplo los tiempos de ejecución de determinada consulta. Para lograr esto no se necesita realizar ninguna modificación a las aplicaciones clientes. (MySQL AB 2008d)

Otra de las facilidades que brinda esta herramienta es que permite hacer una separación transparente de las consultas de escritura y de lectura para que sean enviadas, las primeras (escritura), hacia un servidor *master* de la replicación mysql y, las segundas (lectura), hacia los servidores esclavos de la replicación. De esta manera puede integrarse el cluster al uso de las técnicas de replicación de MySQL y aprovechar las facilidades que brinda. Para que esto ocurra se necesita un *pool* de conexiones. (Kneschke 2007)

Limitaciones de MySQL Proxy

MySQL Proxy sólo es compatible con versiones iguales o superiores a MySQL 5.0. La herramienta se encuentra en estado alpha, es decir que aún no ha sido oficialmente liberada, sino que continua en desarrollo. Por tal motivo todavía atraviesa por una etapa de pruebas y a diario se encuentran algunos errores y *bugs* que deben ser corregidos por sus desarrolladores.

Una vez que MySQL Proxy sea liberada y se encuentre en un estado estable puede constituir una potente solución de cluster de bases de datos para algunas de las aplicaciones que se tratan durante el desarrollo de esta investigación.

1.3.3. MySQL Cluster

(MySQL AB 2008c)

MySQL Cluster es una tecnología para realizar clustering de bases de datos en memoria utilizando una arquitectura que permite el funcionamiento del sistema sin ningún requerimiento especial de *hardware* o *software*. El sistema no tiene ningún punto único de fallo porque cada componente tiene su propia memoria y disco.

MySQL Cluster integra el servidor MySQL estándar con un motor de almacenamiento clusterizado en memoria llamado NDB y está formado por un conjunto de máquinas entre las que se incluyen servidores MySQL, nodos de datos para NDB Cluster, servidores de administración, y algunos programas de acceso a dato. Cada parte del cluster es considerada como un nodo.

Las tablas se almacenan en los nodos de datos y son directamente accesibles desde todos los servidores MySQL en el cluster. Por lo que cualquier cambio realizado en una tabla es visualizado

inmediatamente por todos ellos. El cluster además es tolerante a las fallas que pueden ocurrir en los nodos de datos así como en los nodos servidores de MySQL.

MySQL Cluster proporciona tratamiento de datos clusterizado con alta disponibilidad, alto rendimiento, y escalabilidad disponible.

Limitaciones de MySQL Cluster

Durante el estudio de esta herramienta se realizaron algunas pruebas en el laboratorio. Las principales limitaciones encontradas durante el despliegue de esta solución fueron, en primer lugar, que los nodos de datos consumen una gran cantidad de memoria RAM (*Random Access Memory* / Memoria de Acceso Aleatorio). Esto se debe a que en dichos nodos se almacena un gran volumen de información en memoria, no en disco, que incluye un esquema completo de las bases de datos con los índices de las tablas, datos almacenados, etc.

Lo que significa que para bases de datos de un tamaño considerable grande los recursos de uno de estos nodos del cluster pudieran no ser suficientes. La segunda limitante de gran importancia es que MySQL Cluster utiliza un tipo de *storage engine* denominado NDB que no soporta integridad referencial, por lo que los datos de las tablas pueden quedar inconsistentes luego de realizar alguna operación de modificación sobre las tablas de las bases de datos.

1.3.4. LVS y replicación MySQL

La combinar de las técnicas de balanceo de carga brindadas por LVS y la replicación de datos de MySQL permiten la confección de un cluster de servidores de bases de datos que brinde alta disponibilidad de los servicios.

LVS

Linux Virtual Server (LVS) es una solución para gestionar balanceo de carga en sistemas Linux. Permite la implementación de un cluster de servidores ofreciendo alta disponibilidad y escalabilidad del mismo.

El balanceo de carga se logra a través de *IPVS*, este es un *software* balanceador de carga de capa de trasporte del modelo OSI (*Open Standard Interconection* / modelo de Interconexión de Sistemas Abiertos) de la ISO (*International Standards Organization* / Organización Internacional para la Estandarización) y forma parte del núcleo de Linux. Este *software* puede ser administrado a través de la herramienta *Ipvsadm*. (Martínez Jiménez et al. 2003)

La arquitectura del cluster LVS es completamente transparente para los usuario finales, a pesar de que el cluster está formado por un grupo de máquinas computadoras estas se muestran al cliente como un único servidor. Este aparente único servidor es llamado “servidor virtual” y a los servidores integrantes del cluster se les denomina “servidores reales”.

El nodo encargado del balanceo de carga, conocido como nodo director o balanceador despacha las peticiones a los diferentes servidores y hace que los servicios paralelos del cluster aparezcan como un solo servidor virtual con una sola dirección IP (*Internet Protocol* / Protocolo Internet). LVS permite implementar alta disponibilidad del cluster, para lograr esto se utiliza un nodo de respaldo que asume el balanceo de la carga en caso de que falle el balanceador original, el chequeo del estado del balanceador original se realiza utilizando la herramienta Heartbeat. Por otra parte se puede manejar de forma automática la caída y el restablecimiento de los servidores reales para proporcionar alta disponibilidad de los servicios, lo cual se logra a través del *software* Ldirectord (Ldirectord en epígrafe 1.5). (Martínez Jiménez et al. 2003)

En un cluster LVS el balanceador es un enrutador con las reglas de ruteo modificadas. Cuando una nueva petición de conexión para alguno de los servicios ofrecidos por LVS es recibida, el balanceador la reenviará a alguno de los servidores reales para que sea atendida. La distribución entre los nodos reales se realiza en dependencia de varios algoritmos implementados por LVS. Las conexiones a los servidores de bases de datos se establecen a través del protocolo TCP y estas pueden ser persistentes o no. Cuando existe persistencia se crea una relación entre una dirección IP origen y la dirección IP de un servidor real (destino), esta relación queda registrada en la tabla de rutas que maneja el balanceador garantizando de esta forma que al realizarse una nueva conexión desde la misma dirección IP fuente las peticiones sean enviadas hacia el mismo servidor real (destino). (LVS 1998)

LVS utiliza tres métodos diferentes de reenvío de paquetes (LVS – NAT, LVS – TUN y LVS – DR) los cuales se describen a continuación: (Horman 2003)

LVS - NAT se basa en modificar los campos de dirección IP y puerto destino de los paquetes provenientes de los clientes (IP virtual del balanceador) y reemplazar estos campos por la dirección IP del servidor real escogido para recibir dicha petición, para lograr esto el balanceador y los servidores reales deben pertenecer a la misma red. Las respuestas son enviadas por los servidores reales hacia el cliente a través del balanceador de carga el cual utilizan como *gateway*. Esta técnica tiene la desventaja que pueden producirse cuellos de botella o sobrecargas en el balanceador cuando el cluster está conformado por un número elevado de nodos.

LVS - TUN (*LVS IP Tunneling*) realiza un encapsulamiento de datagramas IP en el nodo balanceador que permite que los paquetes sean enviados directamente hacia los servidores reales o a través de dispositivos intermedios, como enrutadores, atravesando varias redes IP en el camino. Esto hace más flexible la configuración física de la red. Las respuestas son enviadas por los servidores reales directamente hacia los clientes por lo que disminuye la sobrecarga de trabajo en el balanceador y permite la integración de un número mayor de nodos al cluster.

LVS - DR (*LVS Direct Routing*) se basa en realizar cambios en las direcciones MAC de los paquetes para reenviarlos al servidor real elegido manteniendo la dirección IP destino. Los servidores reales pueden aceptar los paquetes IP entrantes ya que tienen un dispositivo con esa dirección IP asignada que es la misma que la del servidor virtual (dirección del servicio virtual). Las respuestas son enviadas desde los servidores reales directamente hacia los clientes. Para que el sistema funcione correctamente los dispositivos de red que tengan asignada la dirección del servicio virtual no deben responder las peticiones ARP, ya que esto ocasionaría conflictos en la red. Por otra parte esta técnica multiplica la escalabilidad del sistema y tiene la ventaja adicional con respecto a IP TUN que libera al balanceador de la carga de trabajo que puede generar el encapsulamiento / desencapsulamiento de los datagramas IP.

Replicación MySQL

La replicación de bases de datos es una funcionalidad que permite que toda acción de modificación de datos (INSERT, UPDATE, DELETE) realizada a un servidor de base de datos se replique automáticamente en otros servidores manteniendo de esta manera una copia exacta de la información en cada servidor.

Esta funcionalidad se basa en un sistema llamado *master - slave* (maestro - esclavo). Su funcionamiento se basa en que todas las ejecuciones realizadas sobre un servidor *master* son leídas y ejecutadas por los servidores esclavos manteniéndose así una copia exacta de la información en todos los servidores. Toda esta operación se realiza de forma asíncrona. Es decir, no hace falta que la conexión entre el esclavo y el *master* esté constantemente activa.

El servidor *master* escribe actualizaciones en el fichero de *log* binario, y mantiene un índice de los ficheros para rastrear las rotaciones de *logs*. Estos *logs* sirven como registros de actualizaciones para enviar a los servidores esclavos. Cuando un esclavo se conecta al *master*, informa al *master* de la posición hasta la que el esclavo ha leído los *logs* en la última actualización satisfactoria. El esclavo recibe cualquier actualización que ha tenido lugar desde entonces, y se bloquea y espera porque se registren nuevas actualizaciones en el *master*. Esto provoca que el esclavo no siempre esté

actualizado, sólo después de leer la última posición del *master* ambos servidores (*master* y esclavo) tendrán los mismos datos. (MySQL AB 2008b)

Un servidor esclavo puede a su vez ser *master* de otro servidor formando así una cadena, aunque se debe tener bien claro donde se actualizan los datos (siempre en el *master*) de lo contrario pueden ocurrir errores fatales en la replicación. También se debe tener en cuenta que un mismo esclavo no puede tener dos servidores *master*.

La replicación es una forma fácil, rápida y económica de incrementar el rendimiento de un sistema y garantizar la integridad de la información. Cuando se usa la replicación y ocurre algún problema en el servidor *master* se puede utilizar el esclavo como copia de seguridad. Por otra parte puede conseguirse un mejor tiempo de respuesta dividiendo la carga de consultas realizadas por los clientes entre los servidores *master* y esclavos aprovechando las técnicas de balanceo de carga que brinda LVS. De esta forma se pueden enviar las consultas tipo SELECT a los servidores esclavos y las consultas de modificación de datos (INSERT, UPDATE, DELETE) hacia el *master* garantizando la sincronización de los servidores y disminuyendo de esta manera la carga de los servidores. Se debe tener en cuenta que esta estrategia de balanceo de carga es efectiva si predominan las consultas que no actualizan datos, caso más habitual para las aplicaciones de tipo web. (Balling et al. 2004)

Con la unión de las facilidades que ofrece el *software* LVS y las ventajas de la replicación MySQL se puede confeccionar un cluster para aplicaciones web cuyos requisitos no funcionales requieran la separación de las consultas de lectura y escritura para que estas sean enviadas a servidores diferentes.

Limitaciones de LVS y la replicación MySQL

Las limitaciones de un cluster conformado por la combinación de técnicas LVS y la replicación MySQL están definidas principalmente según los métodos de reenvío de paquetes utilizados para el balanceo de carga. En el caso de LVS - NAT la mayor limitación se encuentra en que con el uso de esta técnica puede producirse un cuello de botella (*bottleneck*) en el balanceador de carga ya que tanto las peticiones como las respuestas a las mismas son enviadas a través de él como se explicó anteriormente.

La desventaja de utilizar un método IP – TUN es que tanto el director como el servidor tienen que poder crear interfaces de tipo *tunneling*, y como consecuencia de hacer *IP-tunneling* siempre estará implícito un tiempo de procesamiento ocupado en encapsular o desencapsular los paquetes lo cual introduce ciertas demoras en los tiempos de respuesta a las consultas realizadas a los servidores de bases de datos. Por otra parte el uso de una técnica LVS – DR tiene una escalabilidad limitada debido

a que la red sobre la que funciona está limitada a un único segmento ethernet por motivos del direccionamiento mediante ARP.

1.3.5. Sequoia

Sequoia es un middleware transparente para clusters de bases de datos que proporciona balanceo de carga y recuperación ante fallos de casi cualquier RDBMS (*Relational Database Management System*). Forma parte del proyecto Continuent y anteriormente recibía el nombre de C-JDBC. Este cluster permite a cualquier aplicación de Java el acceso transparente a un cluster de bases de datos a través del driver JDBC y sin necesidad de modificar las aplicaciones clientes.

La arquitectura de Sequoia está compuesta por tres componentes fundamentales, ellos son: el controlador, las bases de datos virtuales y los *backends*. (Cecchet et al. 2006)

El **controlador** (*controller*) es un nodo del cluster que se encarga de manejar las bases de datos virtuales y sus respectivos *backends*. Un controlador puede manejar más de una base de datos virtual y por consiguiente más de un *backend*, sin embargo un mismo *backend* no puede ser compartido por dos controladores al mismo tiempo.

Un **backend** es un objeto que utiliza Sequoia para manejar los servidores de bases de datos. Este comprende una instancia de la base de datos (no la base de datos como tal), lo cual quiere decir que si un *backend* es deshabilitado la base de datos permanece operacional. Un *backend* puede ser deshabilitado cuando se realizar la copia de una base de datos, de esta manera se evita que se ejecuten consultas sobre la base de datos durante la ejecución de la copia y se garantiza la consistencia de la base de datos.

Las **bases de datos virtuales** (*virtual database*) representan un grupo de bases de datos reales que se encuentran distribuidas entre los diferentes nodos del cluster, haciéndolas transparentes a los clientes. Cada base de datos tiene un método de autenticación de los usuarios que no necesariamente es el mismo que el que se utiliza para acceder a los RDBMS.

Sequoia maneja el balanceo de dos maneras:

- Balanceando las conexiones de los clientes entre los controladores.
- Balanceando las consultas de lectura (SELECT) entre los *backends*.

El conector de Sequoia establece la conexión con un controlador usando una URL (*Uniform Resouce Locator* / Localizador Uniforme de Recursos) especificada por el cliente, dicha URL contiene una lista de direcciones IP (*Internet Protocol* / Protocolo Internet) de los controladores que contienen la base de

datos virtual a la cual se quiere acceder. La lista de controladores definida en la URL se utiliza para distribuir el balanceo entre cada una de las direcciones IP listadas, esto se hace siguiendo tres políticas diferentes (*random, round-robin, ordered*) que pueden ser especificadas como una opción más de la URL. (Cecchet et al. 2006)

Ordered: las conexiones se realizan en el mismo orden en que se listan los controladores en la URL, se establece una conexión con el primer controlador y si esta falla se conecta al segundo y así sucesivamente, esta opción no realiza un balanceo equitativo de las conexiones entre controladores.

Random: la selección del controlador se realiza de manera aleatoria.

Round Robin: los controladores se seleccionan mediante un ciclo comenzando por el primero y una vez seleccionado el último el ciclo vuelve a comenzar.

Sequoia balancea las consultas de lectura entre los diferentes *backends* según tres métodos de balanceo (least pending requests first, round robin, weighted round robin) (Continuent.org 2007b)

Least Pending Request First: la petición es enviada al *backend* que tiene menor cantidad de peticiones pendientes a dar respuesta.

Round Robin: las peticiones se envían de forma cíclica a todos los *backends*, cuando llega al último vuelve a comenzar por el primero.

Weighted Round Robin: semejante al método anterior pero además se asigna un peso a cada *backends*, el peso determinar en que proporción recibe peticiones un *backend* con respecto a los demás. Un *backend* con peso 2 recibe el doble de peticiones que un *backend* de peso 1.

Las consultas de modificación de datos (INSERT, UPDATE, DELETE) realizadas sobre una base de datos virtual son enviadas por el controlador hacia los *backends* donde se encuentren las tablas que almacenan estos datos en dependencia del método de replicación empleado en la configuración de Sequoia.

La replicación se maneja a través de los tres niveles de RAIDb (*Redundant Array of Inexpensive Databases* / Arreglo Redundante de Bases de datos Baratas) pertenecientes a la arquitectura GORDA (Oliveira 2006). Este concepto es un acrónimo del concepto de RAID (*Redundant Array of Inexpensive Disks* / Arreglo Redundante de Discos Baratos) pero aplicado a bases de datos. Sequoia ofrece tres niveles principales de realizar la replicación: (Cecchet et al. 2003)

RAIDb-0 o particionado completo, se encarga de particionar las tablas de las bases de datos entre varios nodos del cluster, este nivel no ofrece tolerancia ante la ocurrencia de fallos.

RAIDb-1 o replicación completa, como su nombre lo indica cada base de datos estará replicada completamente en cada uno de los nodos del cluster lo cual ofrece alta tolerancia ante la ocurrencia de errores y capacidad de recuperación, además ante el posible fallo de un *backend* el cluster permanece operacional.

RAIDb-2 o replicación parcial, este combina las dos técnicas anteriores y distribuye tablas entre los nodos al mismo tiempo que replica algunas de ellas en al menos dos *backends*. RAIDb-2 reduce el tiempo de realización de consultas de lectura en comparación con RAIDb-1, sin embargo la tolerancia a fallos es menor y presenta algunos problemas en la realización de copias de respaldo y la ejecución de sentencias JOIN entre las tablas distribuidas de las bases de datos.

La utilización de estos niveles de RAIDb puede combinarse para lograr un escalamiento vertical del cluster.

Sequoia implementa un mecanismo de recuperación y sincronización de datos, llamado **recovery log**. Este mecanismo se basa en establecer un punto de chequeo a cada uno de los *backends* una vez que se ha realizado una copia de respaldo de ellos y a partir de este punto se almacenan las actualizaciones realizadas en las bases de datos virtuales. Si se produce la falla de un servidor de base de datos, mientras se encuentra fuera de servicio, las consultas realizadas a los demás servidores se almacenan en el *recovery log*. Posteriormente se puede restablecer la base de datos caída a partir de la copia de seguridad y leyendo todas las modificaciones guardadas después del punto de chequeo. De esta manera la base de datos vuelve a estar actualiza con todas las demás. La información necesaria para realizar la recuperación ante fallos puede ser almacenada en un servidor de base de datos MySQL.

El cluster cuenta también con mecanismos para realizar copias de seguridad. Estos mecanismos, denominados *backups*, garantizan que se puedan agregar nuevos *backends* a la base de datos virtual de manera dinámica, o sea que no se necesita detener y volver a iniciar el sistema para ello. Un *backup* se puede utilizar para restablecer un *backend* cuando ocurren fallos.

Limitaciones de Sequoia

Durante las pruebas realizadas con la herramienta Sequoia se comprobó que una de las mayores limitaciones que presenta es que, a diferencia de otros balanceadores de carga como LVS, consume un por ciento elevado de recursos, sobresaliendo el consumo de CPU. El alto consumo de CPU del controlador Sequoia constituye una gran desventaja de esta solución ya que limita la escalabilidad del cluster.

A pesar de las limitaciones que presenta Sequoia se puede concluir que, por las características que presenta esta herramienta, es posible utilizarla para confeccionar una solución de cluster para las aplicaciones web implementadas en lenguaje Java.

1.3.6. Myosotis y Sequoia

Myosotis es una solución que actúa como *proxy* de Sequoia y permite que las aplicaciones clientes que utilizan servidores de bases de datos MySQL y PostgreSQL puedan establecer una conexión transparente con el cluster Sequoia sin necesidad de efectuar cambios en el código de implementación de la aplicación o en las librerías utilizadas por la misma. Las aplicaciones no necesariamente tienen que estar implementadas en Java. (Continuent.org 2007a)

Myosotis es visto como un servidor MySQL o PostrgeSQL para las aplicaciones clientes ocultando de esta manera la granja de servidores que conforman el cluster. Los servidores de bases de datos reales que forman el cluster puedan ser de cualquier proveedor, no necesariamente MySQL o PostgreSQL y esto se logra gracias a las facilidades que brinda Sequoia.

La configuración de Myosotis es muy sencilla. Para acceder al cluster Sequoia se utiliza el driver JDBC de Sequoia, los clientes acceden a Myosotis utilizando el driver conector a bases de datos MySQL o PostgreSQL.

Con la unión del middleware Sequoia y Myosotis se puede conformar un cluster de bases de datos válido casi para cualquier aplicación de tipo web que acceda desde su lógica de negocio a servidores de bases de datos MySQL o PostgreSQL.

Limitaciones de Myosotis y Sequoia

Las limitaciones de esta solución de cluster están definidas por las limitantes de Sequoia mencionadas en el epígrafe anterior.

1.4. Pruebas para medir el rendimiento del cluster y comprobar su funcionamiento

Para medir el rendimiento de un cluster así como comprobar el funcionamiento correcto del mismo se pueden realizar un grupo de pruebas definidas como:

Pruebas de configuración: Una prueba de configuración tiene el objetivo de comprobar el correcto funcionamiento de las aplicaciones sobre una configuración diferente de *software* o *hardware* que la utilizada en el despliegue original de la aplicación.

Pruebas de rendimiento: Determinan o validan la velocidad, escalabilidad y estabilidad de los sistemas o aplicaciones. Estas pruebas se realizan principalmente para medir los tiempos de respuestas y el comportamiento del consumo de recursos en las aplicaciones.

Este tipo de pruebas puede dividirse en dos subcategorías, pruebas de carga y pruebas de stress. (Meier et al. 2007)

- **Pruebas de carga:** Esta prueba se realiza cuando se espera que la aplicación deba soportar grandes volúmenes de trabajo. Está dirigida a determinar el límite operacional de un sistema o aplicación cuando este se encuentran bajo cargas intensivas de trabajo. Con la realización de esta prueba se busca la ocurrencia de picos producidos por las sobrecargas y que normalmente no ocurren en condiciones normales de despliegue de la aplicación.
- **Pruebas de stress:** Se realizan para comprobar el rendimiento que ofrece el sistema cuando se somete a condiciones más allá de las previstas durante las operaciones de producción. Con esta prueba se pueden detectar hasta que punto pueden aprovecharse los recursos con que cuenta el sistema así como detectar sus deficiencias cuando los niveles de stress son altos, por ejemplo se puede detectar capacidad de memoria insuficiente, falta de espacio en disco, recursos compartidos limitados. Estas pruebas están diseñadas para advertir bajo que condiciones una aplicación puede fallar, cual puede ser la falla y qué indicadores deben ser monitoreados para prevenirla.

Pruebas de *benchmark*: Estas pruebas tienen el objetivo de comparar el comportamiento de los sistemas o las aplicaciones sobre diferentes configuraciones de *software* y *hardware*. La realización de este tipo de prueba consiste en medir el rendimiento de las aplicaciones sobre determinada configuración de *software* o *hardware* y comparar los resultados obtenidos con configuraciones similares.

JMeter (Apache Jakarta Project 2008)

Para la realización de las pruebas de rendimiento se necesitan herramientas generadoras de stress que provoquen la sobrecarga del sistema y permitan realizar mediciones acerca del comportamiento y rendimiento de las aplicaciones. Una herramienta de software libre muy útil en el cumplimiento de este propósito es JMeter.

JMeter es una herramienta de carga para llevar a cabo simulaciones sobre muchos recursos de *software*. Inicialmente fue diseñada para pruebas de stress en aplicaciones web, hoy en día, su arquitectura ha evolucionado no sólo para llevar acabo pruebas en componentes habilitados en

Internet (HTTP), sino además en bases de datos, programas en Perl, y prácticamente cualquier otro medio.

Posee la capacidad de realizar desde una solicitud sencilla hasta secuencias de solicitudes que permiten diagnosticar el comportamiento de una aplicación en condiciones de producción. En este sentido, simula todas las funcionalidades de un navegador (*browser*), o de cualquier otro cliente, siendo capaz de manipular resultados durante una solicitud y reutilizarlos para ser empleados en una nueva secuencia.

1.5. Herramientas para implementar alta disponibilidad del cluster

A continuación se describen un grupo de herramientas de software libre con las cuales se puede implementar alta disponibilidad del cluster.

Heartbeat

Heartbeat es un *software* destinado a mantener servicios en alta disponibilidad bajo Linux. Para ello se requiere ejecutar Heartbeat simultáneamente en un mínimo de dos máquinas. La primera máquina, llamada *master*, es la que ofrece normalmente el servicio. La segunda máquina, llamada esclava, es la que suplanta a la *master* en el caso de que por algún motivo deje de funcionar y consecuentemente de prestar sus servicios. (Linux-HA 2006)

A continuación se describen algunos aspectos del funcionamiento de este *software*: (Robertson 2000)

Heartbeat basa su funcionamiento en el envío y recepción de “latidos”, estos latidos son las señales enviadas por los demonios Heartbeat que corren en ambas máquinas. La diferencia entre el *master* y el esclavo radica en que el *master* es quien tiene la prioridad para ofrecer el servicio. El esclavo pasará a ofrecer el servicio sólo cuando deje de escuchar los latidos del *master* por un período predeterminado de tiempo, entonces supondrá que este ha dejado de funcionar. Tan pronto como el esclavo vuelva a escuchar los latidos del *master* detendrá los servicios que estaba ofreciendo para que el *master* le tome el relevo y vuelva a servirlos.

Ambas máquinas, *master* y esclavo deben estar comunicadas al menos por una tarjeta de red, aunque es recomendable que existan conexiones redundantes para prevenir algún fallo en una de las interfaces.

Heartbeat es capaz de lanzar un servicio y brindarlo a través de la misma dirección IP en dos máquinas diferentes, para hacer esto activa un alias en una interfaz de red determinada denominada IP virtual (virtual IP, VIP) y puede detener el servicio y desactivar la VIP cuando lo necesite. Por la VIP

los clientes acceden al servicio que se presta sin tener conocimiento de cual de los servidores es el que se encuentra en funcionamiento, el *master* o el esclavo.

El proceso mediante el cual Heartbeat maneja el levantamiento de las VIP se denomina *takeover*. Este proceso garantiza que, cuando una de las máquinas deja de ofrecer los servicios que presta, se desactive la VIP para que pueda ser activada en la máquina que la suplanta y de esta manera se evita la ocurrencia de conflictos de IP en la red. El takeover fuerza a las demás computadoras de la red a que actualicen sus tablas ARP, para que los paquetes sean encapsulados con la dirección MAC correcta.

Ldirectord

Ldirectord (*Linux Directord Daemon*) es un demonio para monitorear los servicios ofrecidos en los servidores reales en un cluster LVS. Su funcionamiento consiste en hacer un chequeo constante y verificar la disponibilidad de los servicios que prestan los servidores reales del cluster. Cuando Ldirectord detecta la caída de alguno de los servicios monitoreados en uno de los servidores reales este hace una re configuración automática de la tabla de rutas, utilizando el *software* Ipvsadm, para que el balanceador deje de enviar paquetes hacia este servidor.

Para logra esto en su configuración se debe especificar los servicios que se prestan y los servidores reales asociados a ellos. Los servicios que se pueden monitorear son: ftp, smtp, http, pop, pops, nntp, imap, imaps, ldap, https, dns, mysql, pgsql, sip. (Martínez Jiménez et al. 2003)

1.6. Herramientas para realizar monitoreo

Para asegurar el funcionamiento correcto de las soluciones de cluster de bases de datos implementadas se debe realizar un monitoreo sistemático del funcionamiento del sistema. A continuación se describen algunas herramientas de software libre utilizadas con este propósito.

MRTG

MRTG (*Multi Router Traffic Grapher*) es una herramienta que inicialmente fue creada para representar de forma gráfica el tráfico de red que atravesaba las interfaces de los enrutadores, sin embargo esta se puede usar para representar prácticamente cualquier dato del sistema. MRTG genera páginas html que contienen imágenes relacionadas con los datos del monitoreo. (Oetiker 2008b)

La captura de datos se puede hacer de dos formas, a través de *scripts* o usando SNMP.

SNMP (*Simple Network Management Protocol* / Protocolo Simple de Administración de Red) es un protocolo de la capa de presentación, mantiene y gestiona una base de datos, llamada MIB

(*Management Information Base* / Base de Información de Gestión), dentro de la cual están actualizados cientos de parámetros del sistema. SNMP permite hacer consultas a esta base de datos, e incluso modificar sus valores. (Millán Tejedor)

Además de SNMP para monitorizar valores, se pueden utilizar programas de captura de datos y pasar estos datos a MRTG para que los represente gráficamente. Esto se logra a través de la confección de *scripts* de captura de datos.

MRTG puede integrarse con una herramienta llamada RRDtool (*Round Robin Databases tool*). Esta herramienta, confeccionada por el mismo autor de MRTG, utiliza el mismo motor gráfico y permite almacenar y representar datos, haciendo uso del protocolo SNMP, los cuales almacena en una base de datos circular de tamaño fijo. (Oetiker 2008a)

La herramienta ha sido confeccionada para resolver algunas limitantes que tiene el MRTG en cuanto a rendimiento y flexibilidad en el graficado. Por ejemplo, con el uso de RRDtool el sistema de monitoreo puede ser más rápido ya que MRTG no tiene que generar todos los archivos PNG para los gráficos proceso que le lleva alrededor de cinco minutos, al contrario los gráficos son generados a demanda, es decir cuando el usuario lo solicite.

Nagios

Nagios es un sistema open source muy configurable y popular que se utiliza para monitorizar las computadoras y servicios de una red. Cuenta con un sistema de avisos para alertar a sus contactos cuando ocurre algún problema en los servicios o computadoras que monitorea, igualmente alerta cuando estos problemas son resueltos. Para enviar los avisos de fallo en tiempo real Nagios hace uso de varios medios como por ejemplo e-mail, jabber, SMS, fax y otros programas externos. (Galstad 2007)

Esta herramienta puede monitorizar los servicios SMTP, POP3, HTTP, NTTP, ICMP y SNMP; además monitoriza los recursos de un sistema (carga del procesador, uso de los discos, *logs* del sistema); permite a los usuarios desarrollar sus propios *scripts* de chequeos usando varios lenguajes de programación (Bash, C++, Perl, Ruby, Pitón, PHP, C#, etc.); posee un manejador de eventos que le permite ejecutar determinados programas cuando se detecta un problema. Además brinda una interfaz web donde se muestran algunos datos como el estado actual de la red, notificaciones, historial de problemas, archivos de *logs*, etc. (Galstad 2007)

Sysstat

El paquete Sysstat contiene utilidades para monitorizar el rendimiento del sistema y la actividad del mismo. Sysstat contiene la utilidad *sar*, común en muchos Unix comerciales, y herramientas que se pueden programar vía cron para recoger datos de rendimiento y actividad y mantener un historial. (BLSF Equipo de desarrollo 2005)

Con la herramienta sar se pueden monitorear varios recursos del sistema tales como tasa de transferencia a disco, actividad de la red, memoria RAM y utilización de espacio en SWAP, uso de CPU, actividades del *kernel* de Linux, entre otras.

1.7. Otras herramientas

Un aspecto importante a tener en cuenta durante el despliegue de cualquier sistema es la seguridad del mismo, a continuación se describe una herramienta de software libre muy utilizada con este propósito.

Iptables

El *kernel* (núcleo) del sistema operativo GNU/Linux incorpora el manejo de paquetes TCP/IP. Iptables es una herramienta que forma parte del kernel de Linux a partir de su versión 2.4. la cual permite establecer un grupo de reglas de filtraje para indicarle al *kernel* cómo debe manejar los paquetes TCP/IP.

El manejo de los paquetes se realiza identificando diferentes partes que conforman sus cabeceras TCP (*Transmission Control Protocol*), e IP (*Internet Protocol*). De esta manera se pueden confeccionar reglas de filtraje teniendo en cuenta algunas características que los diferencian entre sí tales como la dirección IP origen y destino de los paquetes, el puerto, tipo de servicio (TOS), entre otras.

Las acciones que el *kernel* puede realizar sobre los paquetes son de aceptación, negación, rechazo, además puede establecer marcas sobre ellos y modificados. (Andreasson 2007)

1.8. Conclusiones

Durante el desarrollo de este capítulo se realizó un estudio de varias soluciones con software libre existentes en el mundo para implementar un cluster de servidores de bases de datos. En el estudio realizado se profundizó en las características de las soluciones así como las limitaciones de cada una de ellas. También se mencionaron herramientas utilizadas en la realización de las tareas de monitoreo, seguridad y alta disponibilidad del cluster. Por otra parte se definió un grupo de pruebas las cuales se realizan con el propósito de validar las soluciones de cluster.

CAPÍTULO 2: ANÁLISIS DE LAS CARACTERÍSTICAS DE LAS APLICACIONES

2.1. Características generales de las aplicaciones

Para el desarrollo y cumplimiento de los objetivos de esta investigación fue necesario realizar un estudio acerca de las aplicaciones web desarrolladas por la Universidad de las Ciencias Informáticas (UCI) y Softel para el sector de la Salud Pública, que actualmente se encuentran desplegadas o en fase de desarrollo.

Este estudio se realizó con el objetivo de conocer las características de las aplicaciones para conformar agrupaciones de las mismas, teniendo en cuenta características comunes entre ellas, así como diseñar una propuesta de solución de cluster de bases de datos válida para cada uno de los grupos conformados. Para obtener la información necesaria se realizaron un grupo de entrevistas a los diseñadores de las aplicaciones así como a los jefes de proyectos de la Facultad 7 y Softel, además se consultaron los documentos de arquitectura de dichas aplicaciones.

En general estas aplicaciones se caracterizan por haber sido diseñadas sobre software libre, son aplicaciones de tipo web cuya arquitectura fue concebida para ser desplegadas de forma centralizada. Además utilizan un servidor de base de datos MySQL en alguna de sus versiones y fueron implementadas utilizando un lenguaje de programación PHP o JAVA. Otra característica significativa de estas aplicaciones es la forma en que se produce el acceso a los datos ya que algunas de ellas fueron diseñadas para prever balanceo de carga un una configuración de replicación *Master* - *Slave* entre servidores MySQL, lo que implica que las consultas de lectura y escritura se realizan en servidores diferentes.

La tabla 2.1 muestra el listado de aplicaciones conformado durante la realización de las entrevistas y algunas de sus características.

Tabla 2.1. Listado de aplicaciones web

Nombre	Servidor de base de datos	Lenguaje de programación	Método de acceso a datos	Uso de transaccio-nes	Uso de procedimientos almacenados	Separación de consultas
Registro de Ciudadanos (RC)	MySQL 4.0	PHP 4	DBx	no	no	si
Registro del Personal de la Salud (RPS)	MySQL 4.0	PHP 4	DBx	no	no	si
Registro de Unidades de Salud (RUS)	MySQL 4.0	PHP 4	DBx	no	no	si
Registro Equipos Médicos (REM)	MySQL 4.0	PHP 4	DBx	no	no	si
Registro Equipos No Médicos (RENM)	MySQL 4.0	PHP 4	DBx	no	no	si
Registro de la Clasificación Internacional de Enfermedades y Problemas Relacionados de Salud (RCIE)	MySQL 4.0	PHP 4	DBx	no	no	si
Registro Problemas de Salud de la Atención Primaria (RPSAP)	MySQL 4.0	PHP 4	DBx	no	no	si
Registro Área de Salud (RAS)	MySQL 4.0	PHP 4	DBx	no	no	si
Registro de Ubicación (RU)	MySQL 4.0	PHP 4	DBx	no	no	si
Registro Servicios Médicos (RSM)	MySQL 4.0	PHP 4	DBx	no	no	si
Registro de estudiantes (REST)	MySQL 4.0	PHP 4	DBx	no	no	si
Registro de localidades (RL)	MySQL 4.0	PHP 4	DBx	no	no	si
Registro de Población (RPOB)	MySQL 4.0	PHP 4	DBx	no	no	si

Nombre	Servidor de base de datos	Lenguaje de programación	Método de acceso a datos	Uso de transacciones	Uso de procedimientos almacenados	Separación de consultas
Componente de seguridad de SiSalud (SAAA)	MySQL 4.0	PHP 4	DBx	no	no	si
Registro de Indicadores y Conductas de la Atención Primaria (RICAP)	MySQL 4.0	PHP 4	DBx	no	no	si
Registro de Enfermedades de Declaración Obligatoria (REDO)	MySQL 4.0	PHP 4	DBx	no	no	si
Registro de Fallecidos (RFALL)	MySQL 4.0	PHP 4	DBx	no	no	si
Registro de Partos y Nacimientos (RPN)	MySQL 4.0	PHP 4	DBx	no	no	si
Registro de Actividades Diarias (RAD)	MySQL 4.0	PHP 4	DBx	no	no	si
Registro de Vacunación (RVAC)	MySQL 4.0	PHP 4	DBx	no	no	si
Registro Centralizado de Donantes (RCD)	MySQL 5.0	Java	JDBC	si	no	no
Componente de seguridad implementado en Java	MySQL 5.0	Java	JDBC	si	no	no
Balance Material	MySQL 5.0	PHP 5	extensión MySQL de PHP	no	si	no
Sistema de Información Estadística Complementaria de Salud	MySQL 5.0	PHP 5	extensión MySQL de PHP	no	si	no
Sistema de gestión de Información en el proceso de formación de Recursos Humanos en Salud.	MySQL 5.0	PHP 5	extensión MySQL de PHP	no	si	no

Estas aplicaciones tienen otras características que no se mencionan en la tabla 2.1 ya que no afectan el funcionamiento de un cluster de bases de datos.

2.2. Estratificación del conjunto de aplicaciones

Como se puede apreciar en la tabla 2.1, muchas de las aplicaciones listadas tienen características semejantes. Estas aplicaciones se agruparon con el objetivo de conformar una solución de cluster válida para cada grupo donde se viera representada cada una de ellas.

Los grupos se definieron teniendo en cuenta principalmente tres características:

- Servidor de base de datos.
- Lenguaje de programación utilizado.
- Separación de consultas de lectura y escritura.

Como resultado se obtuvieron tres grupos de aplicaciones los cuales se describen a continuación.

En el **primer grupo** se encuentran las aplicaciones desarrolladas en lenguaje de programación PHP 4 implementadas para que realicen separación de consultas de lectura y escritura hacia diferentes servidores de bases de datos MySQL versión 4.0 para aprovechar las técnicas de balanceo y replicación de datos. El acceso a los datos se realiza a través del módulo DBx de PHP, estas aplicaciones no realizan transacciones ni maneja procedimientos almacenados. Dentro de este grupo se encuentran las aplicaciones que han sido desarrolladas por el proyecto Atención Primaria a la Salud (APS) y la empresa Softel y comprenden un total de 20 aplicaciones.

Dentro del **segundo grupo** se encuentran dos aplicaciones implementadas en lenguaje de programación Java que utilizan un servidor de bases de datos MySQL versión 5.0 y no realizan separación de consultas de lectura y escritura. El acceso a la capa de datos se realiza a través del driver JDBC, estas aplicaciones no maneja procedimientos almacenados y si realizan transacciones. En este grupo se encuentran las aplicaciones que han sido desarrolladas por la empresa Softel y comprenden un total de 2 aplicaciones.

El **tercer grupo** está compuesto por las aplicaciones implementadas en lenguaje de programación PHP 5, utilizan un servidor de base de datos MySQL versión 5.0 y no separan las consultas de escritura y lectura hacia servidores diferentes. El acceso a los datos se realiza a través de la extensión MySQL para PHP, no realizan transacciones y si manejan procedimientos almacenados. Dentro de este grupo se encuentran las aplicaciones que han sido desarrolladas principalmente por la Facultad 7 de la Universidad de las Ciencias Informáticas (UCI) y comprenden un total de tres aplicaciones.

De cada uno de los grupos conformados se seleccionó una aplicación representativa para la realización de pruebas y validación de las soluciones propuestas. Las aplicaciones seleccionadas fueron:

- Primer grupo: Registro de Unidades de Salud (RUS).
- Segundo grupo: Registro Centralizado de Donantes (RCD).
- Tercer grupo: Balance Material.

Se definió que los resultados obtenidos en el despliegue de cada aplicación representativa sobre una configuración de cluster de bases de datos se hacen extensibles para el resto de las aplicaciones que conforman el grupo al cual representa dicha aplicación, esto es posible ya que todas las aplicaciones agrupadas tienen las mismas características y por tanto su comportamiento es semejante.

2.3. Descripción de las aplicaciones representativas

Para conformar un cluster de base de datos se deben tener en cuenta las características de las aplicaciones que serán desplegadas sobre el cluster para realizar una correcta configuración del mismo y que no se afecten las funcionalidades de las aplicaciones desplegadas. Por tal motivo resulta de vital importancia el conocimiento de la arquitectura de las aplicaciones para diseñar una solución de cluster válida. Las aplicaciones de muestra han sido desarrolladas para que sean desplegadas sobre sistema operativo Linux utilizando una arquitectura basada en componentes. Cada una de ellas cuenta además con un grupo de características que las diferencian entre sí las cuales se describen a continuación.

Descripción de la aplicación Registro de Unidades de Salud (RUS)

La aplicación Registro de Unidades de Salud (RUS) ha sido implementada utilizando una arquitectura de tres capas, una capa para la presentación que contiene la interfaz de usuario, otra que contiene los servicios que dan cumplimiento a los requisitos funcionales del sistema (capa de negocio) y otra para el almacenamiento (capa de datos). Cada una de estas capas puede ser separada en servidores individuales (tres niveles de despliegue), coexistir con otra capa en el mismo servidor (dos niveles de despliegue) por ejemplo presentación y lógica de negocio en el mismo servidor o lógica de negocio y capa de datos en el mismo servidor; incluso pueden coexistir las tres en un mismo servidor (un nivel de despliegue).

Los niveles de despliegue de las capas se aplican en dependencia de las necesidades de la aplicación (arquitectura, seguridad, etc). Cuando se espera que los niveles de procesamiento en una de las capas sean mayores que los del resto puede utilizarse más de un servidor para el despliegue de la misma, de

esta forma, por ejemplo, se puede tener una capa de presentación y lógica de negocio desplegada en dos servidores diferentes y una capa de datos desplegada en varios servidores los cuales recibirán las peticiones del servidor en que resida la capa de lógica de negocio. Ello posibilita que se pueda emplear una técnica de clustering en la capa de datos sin que se afecte la funcionalidad del sistema. (Softel 2006)

Esta aplicación está concebida sobre una arquitectura basada en componentes y orientada a servicios, utiliza PLASER (PLAtaforma de SERvicios), *framework* integrado por varias clases PHP que soporta como llamada RPC (*Remote Procedure Call* / Llamada a Procedimiento Remoto) el protocolo SOAP (*Simple Object Access Protocol* / Protocolo de Acceso a Objetos) y utiliza la capa de abstracción de base de datos DBx, módulo de PHP utilizado para proporcionar el acceso a bases de datos de varios proveedores. (Softel 2006)

La aplicación utiliza un servidor de base de datos MySQL versión 4.0. La información necesaria para su funcionamiento se almacena en dos bases de datos llamadas *BD_UnidadesSalud* y *WF_UnidadesSalud*. Para que los usuarios puedan acceder al sistema, se utiliza un componente de seguridad externo basado en el modelo de Autenticación, Autorización y Contabilidad Única (SAAA por sus siglas en inglés) además consume los servicios del componente Registro de Ubicación.

El lenguaje de programación utilizado en la implementación es PHP 4.

El acceso a los datos se realiza de manera que soporta el balanceo de carga para una configuración de replicación *Master-Slave* entre servidores MySQL. El uso de este requisito no funcional del sistema brinda una mayor confiabilidad ya que el sistema permanece operacional ante la caída da uno de los servidores esclavos de la replicación, además se puede restaurar el servidor caído utilizando una copia de los otros esclavos. Por otra parte el balanceo de carga disminuye los tiempos de respuesta al repartir las peticiones entre varios servidores de bases de datos. Este proceso es completamente transparente para el usuario final.

El envío de consultas se realiza hacia dos servidores diferentes, siendo las consultas de lectura (SELECT) enviadas hacia el balanceador de carga y este a su vez las reparte entre los servidores esclavos de la replicación mientras que las consultas de escritura (INSERT, UPDATE, DELETE) se envían hacia el servidor *Master*. (Softel 2007)

Las direcciones IP de los servidores hacia donde van dirigidas las consultas tanto de lectura como escritura deben estar bien definidas en los archivos de configuración de la aplicación que proporcionan acceso a las bases de datos. La figura 2.1 muestra un ejemplo del fichero *configdb.php* que utiliza el módulo Unidades de Salud para acceder a una de las bases de datos mencionadas anteriormente. Los

parámetros especificados para el Servidor Master son utilizados para establecer la conexión con el *master* de la replicación y los del Servidor Slave para el balanceador de carga.

```
<?php
//Servidor Master
$dbz_master_module = DBX_MYSQL;
$dbz_master_host = '10.128.3.32';
$dbz_master_user = 'user';
$dbz_master_passw = 'password';
$dbz_master_db = 'BD_UnidadesSalud';

//Servidor Slave
$dbz_slave_module = DBX_MYSQL;
$dbz_slave_host = '10.128.3.33';
$dbz_slave_user = 'user';
$dbz_slave_passw = 'password';
$dbz_slave_db = 'BD_UnidadesSalud';
?>
```

Figura 2.1. Archivo de configuración para el acceso a datos.

Descripción de la aplicación Registro Centralizado de Donantes (RCD)

El Registro Centralizado de Donantes (RCD) fue desarrollado en la herramienta Genexus, esta herramienta permite el desarrollo de *software* desde múltiples plataformas, ya sea de sistema operativo, lenguajes de programación o motores de bases de datos. La aplicación fue implementada en lenguaje Java y usa un servidor de base de datos MySQL 5.0. Utiliza una arquitectura de tres capas (presentación, lógica de negocio y capa de datos).

La capa de presentación y la de lógica de negocio se despliegan sobre dos servidores de aplicación Apache Tomcat respectivamente, el acceso a la capa de datos se realiza a través del driver JDBC (*Java Database Connectivity*), driver que permite la ejecución de operaciones sobre bases de datos desde el lenguaje Java independientemente del sistema operativo donde se ejecute o de la base de datos a la cual se accede. Específicamente se utiliza el driver conector de JDBC para MySQL.

La información se encuentra almacenada en una base de datos llamada *bs,* además consume los servicios del componente para la seguridad diseñado para la Autenticación, Autorización y Contabilidad Única de los clientes (SAAA).

La aplicación maneja transacciones en las consultas que realiza a las bases de datos, esta característica debe ser tenida en cuenta cuando se despliega la aplicación sobre una configuración de *software* que implique el balanceo de carga entre los servidores de bases de datos MySQL, ya que se pueden producir pérdidas de estado de la transacción realizada cuando se balancean las consultas de la misma transacción hacia servidores de bases de datos diferentes. Lo cual no quiere decir que la aplicación no soporte el balanceo de carga sino que este se debe realizar teniendo en cuenta que las consultas de una misma transacción se ejecuten todas en el mismo servidor de base de datos.

Descripción de la aplicación Balance Material (BM)

La aplicación Balance Material está diseñada utilizando el patrón arquitectónico Modelo Vista Controlador (MVC) el cual define la organización del modelo de diseño en capas lógicamente distribuidas donde cada componente de una capa sólo puede hacer referencia a componentes en capas inmediatamente inferiores. Esta aplicación utiliza una arquitectura en capas que pueden ser desplegadas en dos niveles, dedicando un servidor a la capa de presentación y lógica de negocio y otro para el almacenamiento de los datos (capa de datos).

Para la implementación de la aplicación se utilizó el lenguaje de programación PHP 5, el acceso a la capa de datos desde la lógica de negocio se realiza a través de un grupo de funciones que proporciona el API (*Application Programming Interface* / Interfaz de Programación de Aplicaciones) de PHP para acceder a bases de datos MySQL.

La información se almacena en una única base de datos llamada *db_bm* para ello se utiliza como gestor de base de datos un servidor MySQL versión 5.0.

Una característica importante de esta aplicación es que utiliza procedimientos almacenados, además hace uso de funciones PHP que abren conexiones persistentes a los servidores de bases de datos.

2.4. Conclusiones

Las aplicaciones desarrolladas para informatizar el sector de la salud en Cuba, cuya implementación a estado a cargo de la Universidad de las Ciencias Informáticas (UCI) y la empresa Softel, poseen un grupo de características comunes en cuanto a despliegue y arquitectura. Estas características permiten la implementación de un cluster de servidores en la capa de datos de dichas aplicaciones.

Para proponer una solución de cluster válida para cada aplicación las mismas se separaron en tres grupos y se seleccionó una aplicación representativa de cada grupo. Sobre esta aplicación representativa se realizará el despliegue del cluster y se efectuarán las pruebas para validar la solución de cluster propuesta.

Los resultados de las pruebas realizadas, sean negativos o positivos, serán considerados valederos para el resto de las aplicaciones del grupo al cual pertenece la aplicación probada, esto es posible ya que todos los miembros del grupo tienen las mismas características por lo que su comportamiento sobre el cluster debe ser semejante.

CAPÍTULO 3: PROPUESTA DE DISEÑO DE LAS SOLUCIONES DE CLUSTER

3.1. Soluciones propuestas

Para confeccionar las propuestas de soluciones de cluster de base de datos para las aplicaciones desplegadas o en desarrollo del Ministerio de Salud Pública (MINSAP) se realizó un estudio de varias herramientas, uno de los requisitos que se tuvo en cuenta para la selección realizada fue la búsqueda de soluciones de software libre.

El software libre ofrece ciertas ventajas sobre el software propietario que constituyen un fuerte fundamento por el cual se realizó la selección de herramientas que cumplieran con esta característica, algunas de estas ventajas se mencionan a continuación: (Culebro Juárez et al. 2006)

Bajo costo de adquisición y libre uso: Las herramientas libres brindan la posibilidad a todos los usuarios que las utilizan a hacerlo sin necesidad de pagar una licencia de uso. Además, como su nombre lo indica, el código fuente de las herramientas es de libre distribución lo que le brinda la posibilidad al usuario de personalizar el *software* y adaptarlo a sus condiciones de trabajo.

Requisitos de hardware menores: Las soluciones de software libre tienen unos requisitos de *hardware* menor, y por lo tanto son más baratas de implementar. Por ejemplo, los sistemas Linux que actúan de servidores pueden ser utilizados sin la interfaz gráfica, con la consecuente reducción de requisitos de *hardware* necesarios.

Sistemas sin puertas traseras y más seguros: El acceso al código fuente permite que tanto hackers como empresas de seguridad de todo el mundo puedan auditar los programas, por lo que la existencia de puertas traseras en los programas de software libre es ilógica.

Para conformar la propuesta de cluster de servidores de bases de datos se realizó el estudio de las siguientes soluciones:

- SQL Relay
- MySQL Proxy
- MySQL Cluster
- *Linux Virtual Server* (LVS) y replicación de MySQL
- Sequoia
- Myosotis y Sequoia

Sin embargo, una vez analizadas las características de las aplicaciones listadas en el capítulo 2 de esta investigación y teniendo en cuenta las limitaciones que presentan algunas de las soluciones de cluster estudiadas se decidió proponer tres soluciones de cluster de servidores de bases de datos, ellas son:

- Linux Virtual Server (LVS) y replicación de MySQL
- Sequoia
- Myosotis y Sequoia

Cada una de estas soluciones se puede implementar en uno de los tres grupos de aplicaciones confeccionados. En la tabla 3.1 se muestra la relación entre los tipos de aplicaciones y la solución de cluster que se propone para cada uno de ellos.

Para un mejor entendimiento de la tabla 3.1 Se debe recordar las características de cada grupo confeccionado.

Primer grupo: Lenguaje de programación PHP 4, servidor de base de datos MySQL 4.0, separación de consultas de lectura y escritura, acceso a datos a través del módulo DBx. Muestra: Registro de Unidades de Salud (RUS).

Segundo grupo: Lenguaje de programación Java, servidor de base de datos MySQL 5.0, manejo de transacciones, acceso a dato a través de JDBC. Muestra: Registro Centralizado de Donantes (RCD).

Tercer grupo: Lenguaje de programación PHP 5, servidor de base de datos MySQL 5.0, uso de procedimientos almacenados, acceso a datos a través de la extensión MySQL de PHP. Muestra: Balance Material (BM).

Tabla 3.1. Propuestas de soluciones contra aplicaciones.

Solución de cluster	**Grupo de la aplicación**		
	Primer grupo (RUS)	**Segundo grupo (RCD)**	**Tercer grupo (BM)**
LVS y replicación MySQL	Propuesta	No	No
Sequoia	No	Propuesta	No
Myosotis y Sequoia	Propuesta	No es necesaria	Propuesta

Como se puede apreciar en la tabla 3.1, la solución Myosotis y Sequoia no es necesaria en el caso de las aplicaciones del Segundo grupo, esto se debe a que las aplicaciones de este grupo están

implementadas en Java y establecen una conexión directa con Sequoia sin necesidad de utilizar Myosotis como intermediario.

3.2. Diseño de las soluciones

Se confeccionaron cuatro propuestas de diseño para conformar un cluster de servidores de bases de datos de acuerdo a la tabla 3.1. Se propone que cada solución sea implementada sobre el sistema operativo GNU/Linux distribución Debian 4.0. El despliegue del cluster se realiza en la capa de datos de la aplicación.

Los nodos utilizados para conformar el cluster pueden ser computadoras personales con características de *hardware* variables, sin embargo se recomienda que la velocidad de las tarjetas de red sea al menos de 100 Mbps para lograr un mejor rendimiento del sistema. El cluster debe ser desplegado en un mismo segmento de red localizado en una zona asegurada por las reglas de filtraje de un cortafuego (*firewall*). Además la ubicación física del mismo debe realizarse en una instalación con buena climatización, respaldo eléctrico por si ocurren fallas de energía y acceso restringido al personal.

En los siguientes epígrafes se describe la arquitectura de cada solución propuesta.

3.2.1. Diseño de la solución LVS y replicación MySQL para el primer grupo de aplicaciones

La propuesta de solución de cluster de base de datos LVS y replicación MySQL puede ser aplicada para las soluciones pertenecientes al primer grupo. Estas aplicaciones tienen la característica de que el acceso a datos se realiza de manera que las consultas de lectura y escritura estén dirigidas hacia servidores de bases de datos diferentes.

El cluster está compuesto por (ver figura 3.1):

- Un servidor *master* para la replicación de MySQL.
- Un balanceador de carga.
- Un balanceador de carga de respaldo.
- Un grupo de servidores de bases de datos MySQL, como mínimo dos.

La configuración de la aplicación para que envíe las consultas de lectura y escritura hacia servidores diferentes se realiza en los ficheros correspondientes que proporcionan las direcciones de acceso a las bases de datos. En estos ficheros se debe especificar la dirección IP correspondiente del balanceador de carga hacia donde serán enviadas las consultas de lectura y la dirección IP del servidor *master* de la replicación donde se ejecutarán las consultas de escritura. El balanceador de carga se encargará de

repartir las peticiones entre cada servidor de base de datos disminuyendo así la carga de cada uno de ellos. Mientras que cada acción de modificación de datos realizada en el servidor *master* de la replicación se ejecutará automáticamente en cada uno de los servidores reales, también denominados *esclavos* de la replicación, manteniendo de esta manera todos los servidores sincronizados con la misma información.

Se debe tener estricto cuidado en no realizar una consulta de modificación de datos directamente en uno de los servidores reales ya que esto afectaría la replicación de los datos y por consiguiente la información almacenada en cada servidor no sería la misma. Para que esto no ocurra se debe verificar que todas las consultas de modificación de datos realizadas sobre los servidores esclavos sean el resultado de la replicación de las consultas que se ejecutan en el servidor *master*. Esta operación se puede verificar haciendo un chequeo de las trazas (*logs*) en cada uno de los servidores esclavos y en el *master*.

El balanceador de carga de respaldo constituye una solución de alta disponibilidad que se brinda al cluster. En caso de que, por algún motivo, se produzca la caída del balanceador de carga este nodo de respaldo asumirá su papel instantáneamente. Para los usuarios que navegan en la replicación este proceso es casi transparente. Una vez restablecido, el balanceador original vuelve a tomar el control y el nodo de respaldo queda en estado de espera por si ocurre una nueva caída.

La figura 3.1 muestra la arquitectura general del cluster propuesto.

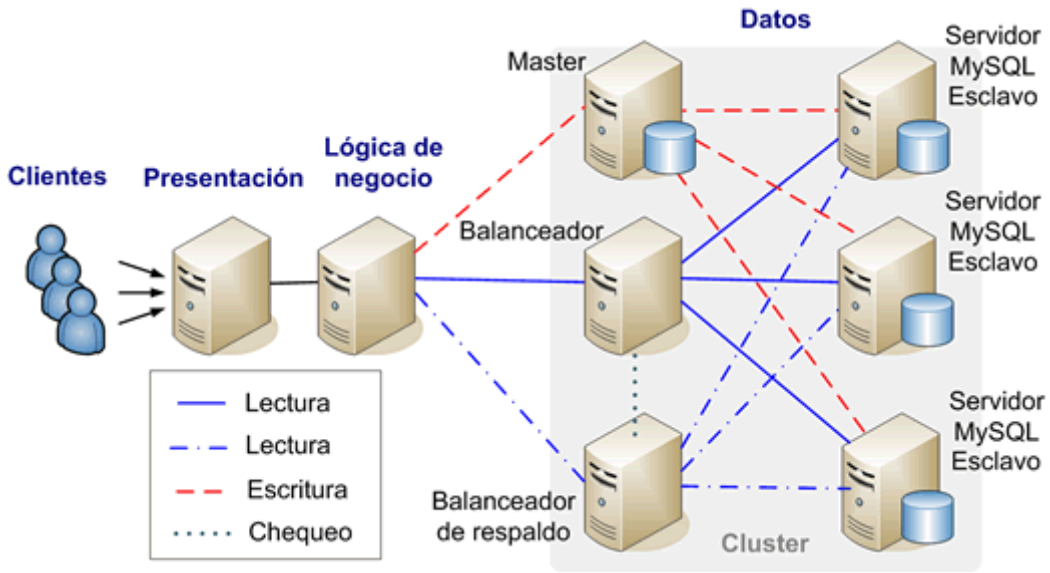

Figura 3.1. Arquitectura del cluster LVS y replicación MySQL.

Para la implementación del sistema se necesitan las siguientes herramientas:

- Ipvsadm (versión utilizada 1.24)
- Ldirectord (versión utilizada 1.2.5)
- Servidor de base de datos MySQL (versión utilizada 4.0.1)
- Heartbeat (versión utilizada 1.2.5)

Para comprender mejor la arquitectura del cluster este se dividirá en dos partes, primeramente se analizará la arquitectura referente al balanceo de carga y luego se observará la replicación de datos.

Balanceo de carga

El cluster está conformado por un nodo que es el encargado de balancear las peticiones entre los servidores reales, en este nodo, llamado balanceador, se instalan las herramientas Ipvsadm y Ldirectord. La herramienta Ipvsadm será la encargada de proporcionar el balanceo mientras que Ldirectord garantizará un mecanismo de recuperación ante la caída de alguno de los servidores reales.

Cuando se realiza una petición al balanceador de carga, este redirecciona la petición hacia una de los servidores reales utilizando una técnica de balanceo vía IP Direct Routing (enrutado directo vía IP), se decidió la utilización de este método ya que con el uso del mismo la carga del balanceador disminuye en comparación a los otros métodos existentes (LVS en epígrafe 1.5) además todos los nodos del cluster pertenecer a una misma subred, característica que facilita el uso de este método.

La distribución de la carga entre los servidores de bases se realiza utilizando el algoritmo menos conexiones con peso (*weigthed least connection*). Se decidió escoger este algoritmo ya que el mismo permite distribuir más peticiones a unos balanceadores que a otros, en caso de que el cluster esté conformado por servidores con diferentes características de *hardware* que impliquen un mayor procesamiento de algunos servidores con respecto a los demás. Si todos los servidores tuvieran el mismo nivel de procesamiento igualmente se puede distribuir la misma cantidad de peticiones entre cada uno de ellos asignando al mismo peso (un valor entero diferente de 0) a todos por igual.

Para chequear el estado de los servicios prestados por los servidores y verificar si estos se encuentran disponibles se utilizó el *software* Ldirectord. La configuración de Ldirectord se realiza en un fichero llamado *ldirectord.cf* el cual debe colocarse en el directorio /etc/ha.d/, en él se especifican las direcciones de los servidores reales, el servicio que se chequea (MySQL), el algoritmo de distribución de carga y la dirección virtual (VIP) del balanceador, entre otros parámetros. La figura 3.2 muestra un fragmento de la configuración que se realizó en este fichero.

```
virtual=10.128.3.240:3306
      real=10.128.3.30:3306 gate 1
      real=10.128.3.31:3306 gate 2
      real=10.128.3.32:3306 gate 2
      service=mysql
      checktype = negotiate
      login = "user"
      passwd = "password"
      database = "database"
      request = "SELECT * FROM database. table"
      scheduler=wlc
      protocol=tcp
```

Figura 3.2. Ejemplo de configuración de Ldirectord.

El balanceador de carga debe tener al menos dos direcciones IP, una de ellas conocida por los clientes y por donde se brinda el servicio que presta el cluster, denominada IP virtual (virtual IP, VIP) y la otra para la comunicación con los nodos del cluster, esta última se definió como un alias de la interfaz de red principal del balanceador. Para garantizar el funcionamiento correcto del balanceador y definir la VIP se utilizó un script del cual se muestra un fragmento en la figura 3.3.

```
echo "0" > /proc/sys/net/ipv4/ip_forward
echo "1" > /proc/sys/net/ipv4/conf/all/send_redirects
echo "1" > /proc/sys/net/ipv4/conf/default/send_redirects
echo "1" > /proc/sys/net/ipv4/conf/eth0/send_redirects
ifconfig eth0:0 10.128.3.240 netmask 255.255.255.255 broadcast 10.128.3.240 up
route add -host 10.128.3.240 dev eth0:0
```

Figura 3.3. Configuración de variables en el balanceador.

Por su parte los servidores de bases de datos deben tener además de su dirección IP real (real IP, RIP) un alias de la interfaz *loopback* con la misma dirección IP que la asignada en el alias del balanceador, esta interfaz no debe responder a las peticiones ARP (*Address Resolution Protocol*/ Protocolo de Resolución de Direcciones) del enrutador de la subred en la que se encuentran ubicados. Para lograr esto es necesario configurar algunas variables del *kernel* (núcleo) de Linux. La configuración de estas variables se realizó mediante un script del cual se muestra un fragmento en la figura 3.4.

```
echo "0" > /proc/sys/net/ipv4/ip_forward
echo "1" > /proc/sys/net/ipv4/conf/all/arp_ignore
echo "2" > /proc/sys/net/ipv4/conf/all/arp_announce
echo "1" > /proc/sys/net/ipv4/conf/eth0/arp_ignore
echo "2" > /proc/sys/net/ipv4/conf/eth0/arp_announce
ifconfig lo:0 10.128.3.240 netmask 255.255.255.255 broadcast 10.128.3.240 up
route add -host 10.128.3.240 dev lo:0
```

Figura 3.4. Configuración de variables en los servidores reales.

La figura 3.5 muestra una configuración más detallada de la arquitectura del balanceador de carga y los servidores reales del cluster.

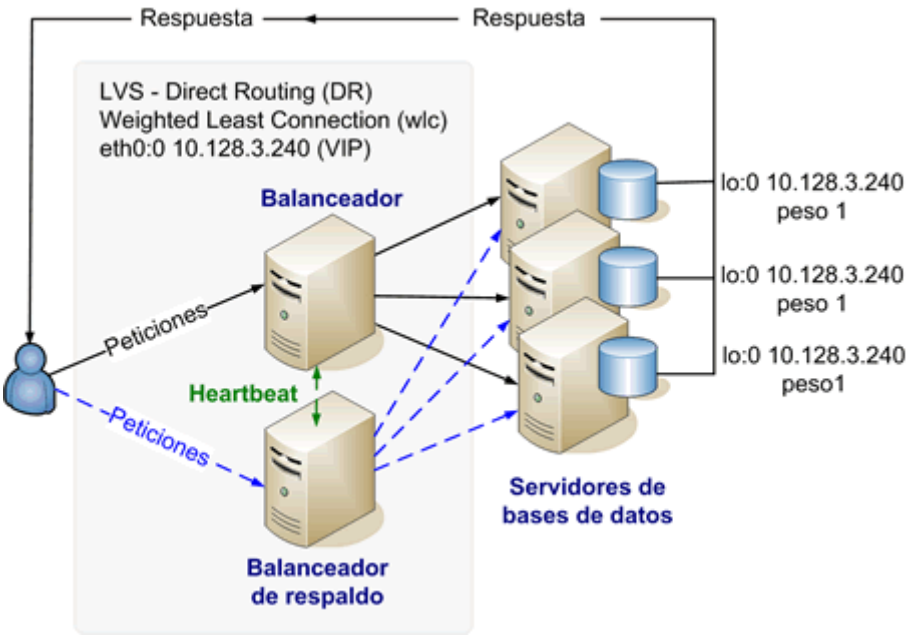

Figura 3.5. Balanceador de carga en el cluster.

Como se puede apreciar en la figura 3.5, la solución de cluster incluye un mecanismo para brindar alta disponibilidad al balanceador de carga. Para ello existe un nodo de respaldo que asume el papel de balanceador de carga cuando se produce la caída del balanceador original, esto se logra gracias a las facilidades que brinda el *software* Heartbeat (Heartbeat en epígrafe 1.6) el cual debe ser instalado en ambos balanceadores, la configuración de este *software* y de la solución en general se puede ver en el documento adjunto *"Instalación y configuración de la solución LVS y replicación MySQL para RUS.pdf"*.

La figura 3.6 y 3.7 muestran un ejemplo de los ficheros ha.cf y haresources que se utilizaron para configurar Heartbeat.

```
debugfile /var/log/ha-debug
logfile /var/log/ha-log
logfacility local0
keepalive 2
deadtime 30
warntime 20
initdead 30
ucast eth0 10.128.3.235
auto_failback on
node node-04 test235
```

Figura 3.6. Fichero ha.cf.

```
node-04 IPaddr::10.128.3.244/32 hadr-master ldirectord
```

Figura 3.7. Fichero haresources.

Replicación de datos

La replicación de datos se realiza desde un nodo llamado *master* sobre el cual se ejecutan todas las consultas de modificación de datos. En este nodo se instaló un servidor de base de datos MySQL versión 4.0.1. Para que la replicación se lleve a cabo se definieron algunos parámetros en el fichero de configuración de MySQL llamado my.cnf el cual se localiza en el directorio /etc/. Los parámetros definidos en este fichero son: un identificador para el *master*, que debe ser un valor único, y además se activa el *log* binario para que las trazas de las consultas realizadas se guarden en el fichero especificado y luego puedan ser leídas por los servidores esclavos de la replicación, de esta manera las bases de datos de los servidores esclavos se mantienen sincronizadas a las del *master*.

Por otra parte, los servidores sobre los cuales se replica la información, denominados esclavos (*slaves*) también requieren de un servidor de base de datos MySQL 4.0.1, sin embargo la configuración del fichero my.cnf cambia. Para cada esclavo se asigna un identificador que debe ser diferente al de todos los demás, también se debe especificar la dirección IP del nodo donde se encuentra el *master* así como una contraseña y un usuario de la replicación para acceder a él. Para definir un usuario de la replicación en el servidor *master* se utiliza el comando GRANT, como se muestra en la figura 3.8.

```
grant replication slave on *.* to reply@'%' identified by 'pass';
flush privileges;
```

Figura 3.8. Creación de un usuario para la replicación.

La estructura de la replicación en el cluster se puede apreciar de manera más detallada en la siguiente figura.

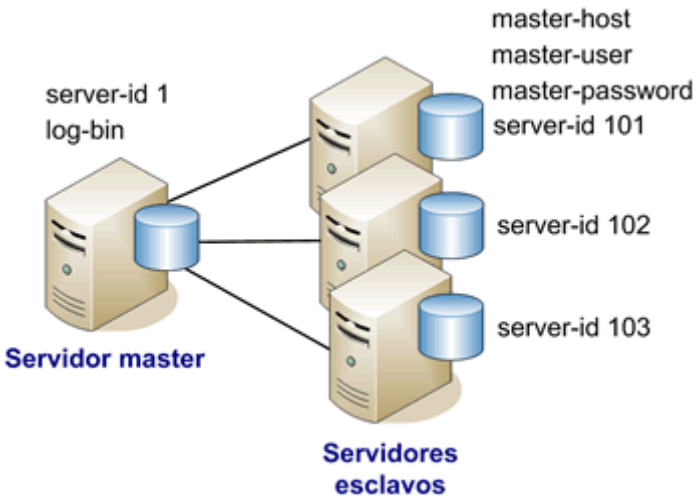

Figura 3.9. Replicación de datos en el cluster.

Recuperación ante fallas del servidor *master*

La propuesta no incluye un mecanismo de recuperación ante fallos del servidor *master* de la replicación, sin embargo, se proponen una serie de pasos a seguir ante una situación como esta:

Para chequear que el sistema funcione correctamente se debe realizar un constante monitoreo de cada uno de los servidores del cluster así como del cluster en general, este proceso se describe en próximos epígrafes. Cuando se detecta, a través del monitoreo, que el servidor *master* ha dejado de prestar sus servicios, el administrador del sistema debe realizar una configuración manual de un *master* sustituto. Este nuevo *master* puede ser el mismo servidor caído, o uno de los nodos esclavos del cluster. Cuando se puede restablecer el servicio en el mismo servidor caído la solución es simple, basta con reiniciar el servicio mysql para que lo replicación continúe sin problemas. Sin embargo cuando no es posible restablecer el servidor *master* original se debe configurar uno de los servidores esclavos para que asuma su papel. Todo este proceso se realiza de forma manual siguiendo los siguientes pasos:

Pasos a seguir en el servidor esclavo que se va a migrar a *master*.

1. Parar el servicio MySQL.
2. Copiar el fichero de configuración de un servidor *master* para la dirección /etc/my.cnf.
3. Establecer el *server-id*, puede ser el mismo que tenía antes como esclavo.

4. Levantar un alias con la dirección IP del antiguo servidor *master*.

```
ifconfig <interfaz de red> <dirección IP> netmask <máscara de red>
broadcast <dirección de broadcast> up
```

5. Iniciar el servicio MySQL.

6. Bloquear tablas.

```
mysql> flush tables with read lock;
```

7. Borrar *log* binarios y reiniciar posición.

```
mysql> reset master;
mysql> quit;
```

8. Hacer un *backup* de todas las base de datos.

```
/usr/local/mysql/bin/mysqldump --master-data --add-locks --lock-tables --
add-drop-table -A -u root -p > /home/dump-data.sql
```

9. Desbloquear tablas.

```
mysql> unlock tables;
```

10. Copiar *backup* para el directorio /home del resto de los servidores esclavos.

Pasos a seguir en los demás servidores esclavos.

11. Parar el servicio MySQL.

12. Iniciar MySQL sin que se levante como esclavo.

```
/usr/local/mysql/bin/mysqld --skip-slave-start &
```

13. Restaurar *backup*.

```
/usr/local/mysql/bin/mysql -u root -p < /home/dump-data.sql
```

14. Reiniciar posición del *log* binario.

```
mysql> reset slave;
```

15. Comenzar la replicación.

```
mysql> start slave;
```

16. Reiniciar el servicio MySQL.

3.2.2. Diseño de la solución Sequoia para el segundo grupo de aplicaciones

La propuesta de solución para las aplicaciones del segundo grupo incluye un mecanismo de balanceo de carga y replicación de datos, ambos se ejecutan desde un mismo nodo al cual van dirigidas todas las peticiones de las aplicaciones clientes. El cluster se compone además de un conjunto de servidores de bases de datos sobre los que se realiza el balanceo y se replica la información. En cada uno de estos servidores se instaló un servidor de base de datos MySQL versión 5.0.3.

El balanceo y la replicación de datos se logran a través del *software* Sequoia el cual se instaló en el nodo balanceador, también llamado controlador. La versión de Sequoia usada en el despliegue fue la 2.10.10, para utilizar esta herramienta fue necesaria la instalación de un kit de desarrollo de *software* java (*Java Software Development Kit*), se recomienda utilizar una versión 1.4.2 o superior. En el nodo controlador también se instaló un servidor de base de datos MySQL 5.0.3 para el almacenamiento de las datos de recuperación del balanceador (Sequoia en epígrafe 1.3).

Para establecer la conexión del controlador con los servidores de bases de datos se utiliza el driver de java para la conexión con MySQL (mysql-connector-java-5.0.4). Sequoia recibe las conexiones de los clientes por el puerto 25322 y esta se establece a través del driver Sequoia, que es una modificación del driver JDBC.

El diseño general del cluster se puede apreciar en la figura 3.10.

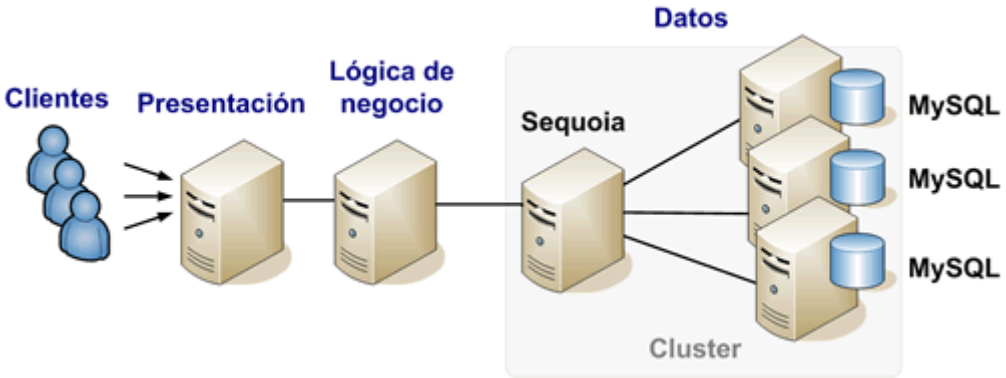

Figura 3.10. Propuesta de diseño para el cluster Sequoia.

El diseño propuesto incluye un controlador que maneja dos bases de datos virtuales, cada una de estas bases de datos virtuales mantiene un *pool* de conexiones con cada *backend*, el tamaño del *pool* es variable en dependencia de la cantidad de conexiones establecidas y se debe fijar un límite para no sobrecargar los servidores de bases de datos. El límite se define teniendo en cuenta la concurrencia que debe soportar el cluster.

Mecanismos de balanceo y replicación

Para repartir las consultas de lectura entre los servidores del cluster se utilizó un algoritmo de balanceo Round Robin ya que es un algoritmo sencillo que garantiza un reparto equitativo entre cada servidor de base de datos. Además proporciona una mayor rapidez en la obtención de los resultados de las peticiones realizadas cuando se tiene una alta concurrencia de usuarios navegando en la aplicación. Otro de los algoritmos probados, el Least Pending Requests First, produjo un retardo en la devolución de las consultas. Tampoco se utilizó Weighted Round Robin ya que la propuesta se diseñó para que todos los servidores de bases de datos tuvieran las mismas características de *hardware*. En caso de que no fuera así este sería el algoritmo a utilizar ya que permite enviar más peticiones a los servidores con mayor capacidad de procesamiento.

La replicación de datos se realizó utilizando el esquema de distribución de datos RAIDb1 que proporciona la arquitectura GORDA. Se escogió este esquema ya que proporciona una mayor rapidez en la realización de las consultas de lectura, este tipo de consultas representan un mayor por ciento con respecto a consultas de escritura realizadas por las aplicaciones para las cuales se propone esta solución. Además RAIDb1 permite mantener una copia exacta de la bases de datos en cada uno de los nodos de datos del cluster, lo cual garantiza que el sistema permanezca operacional ante la ocurrencia de fallos, aún cuando se encuentre un solo servidor de bases de datos prestando sus servicios.

Cuando se utiliza un método RAIDb1 las consultas de modificación de datos son enviadas a todos los servidores de bases de datos del cluster, sin embargo no todos los servidores envían al mismo tiempo la respuesta de que la consulta fue ejecutada exitosamente. Para devolver las respuestas ante una consulta de modificación de la manera más rápida posible se utilizó una política “first” (primero). Esta política envía una respuesta inmediatamente que se ejecuta la consulta en al menos una de los servidores, a diferencia de las otras políticas existentes las cuales esperan que las consultas sean ejecutadas en todos los servidores o al menos en la mitad de ellos para enviar una respuesta.

La estructura del cluster y los componentes del controlador Sequoia pueden apreciarse con más detalle en la figura 3.11.

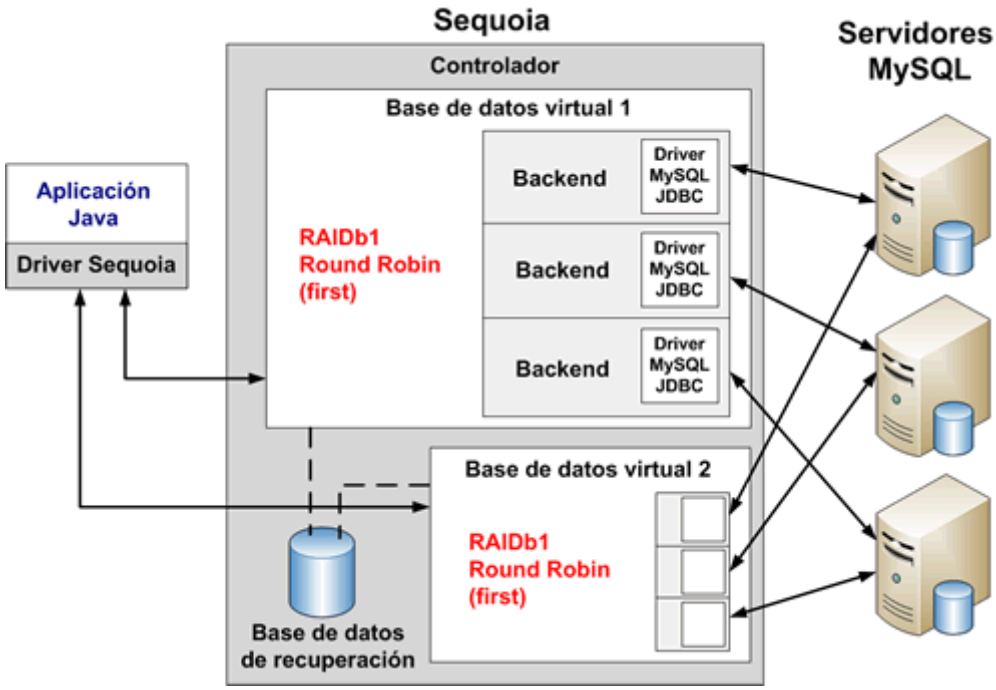

Figura 3.11. Arquitectura del controlador Sequoia.

La configuración de Sequoia se realiza en varios ficheros localizados en el directorio /config/ perteneciente a un grupo de directorios creados durante la instalación del *software*. Cada controlador debe tener un fichero de configuración en el nodo donde se despliegue, además se debe definir una configuración para cada base de datos virtual manejada por el controlador. La configuración del controlador es sencilla, mientras que cada base de datos virtual requiere de un grupo de parámetros para garantizar el buen funcionamiento del cluster. Los parámetros para configurar Sequoia así como los pasos para su instalación adaptados a este diseño se describen en el documento adjunto *"Instalación y configuración de la solución Sequoia para RCD.pdf"*.

3.2.3. Diseño de la solución Myosotis y Sequoia para el primer grupo de aplicaciones

Esta propuesta de solución posee características comunes a la solución anterior. Su despliegue incluye un nodo para el balanceo y la replicación de datos en el cual se debe instalar el *software* Sequoia (versión 2.10.10) y las herramientas necesarias para su correcto funcionamiento (herramientas de desarrollo de java, *Java Software Development Kit* en versiones iguales o superiores a la 1.4.2 y el driver conector de MySQL, se utilizó la versión mysql-connector-java-5.0.4 para la conexión con servidores MySQL en versiones anteriores a la 5.0).

Los nodos de datos del cluster están conformados por un conjunto de servidores de bases de datos MySQL versión 4.0.1.

Las aplicaciones para las cuales se propone la solución de cluster no pueden establecer una conexión directa con el cluster Sequoia ya que no utilizan el driver JDBC para acceder a la capa de datos. Por tal motivo se utilizó el *software* Myosotis (versión 0.6.1) que actúa como intermediario para facilitar el acceso de la aplicación al cluster.

Sequoia y Myosotis coexisten en un mismo nodo del cluster aunque se pueden desplegar de forma separada, sin embargo se decidió hacerlo de esta manera ya que Myosotis es un middleware muy sencillo que no consume grandes cantidades de recursos. Por otra parte se evita la utilización de un nodo adicional en el cluster de manera innecesaria. El despliegue de la solución propuesta se puede apreciar en la figura 3.12.

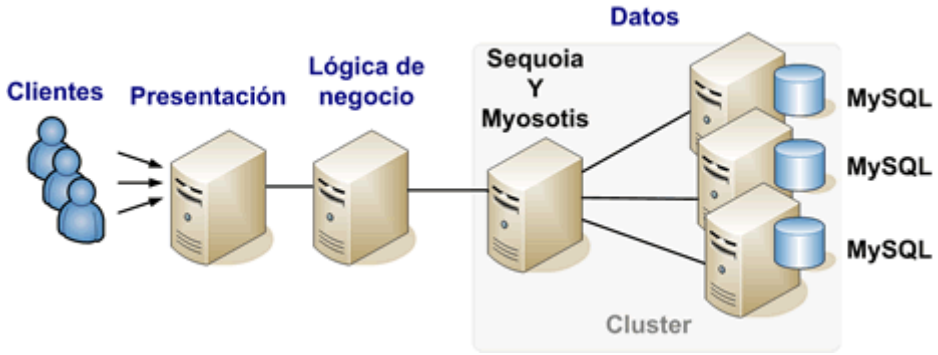

Figura 3.12. Propuesta de diseño del cluster Sequoia y Myosotis (primera variante).

Modificaciones en la instalación del servidor de lógica de negocio

Para que se pueda establecer el acceso de la aplicación al middleware Myosotis, es necesario realizar algunas variaciones en el proceso de instalación del servidor de lógica de negocio. La instalación debe realizarse según los pasos descritos en el documento de despliegue de la aplicación Registro de Unidades de Salud (RUS), teniendo en cuenta las siguientes especificaciones:

- Se utilizarán nuevas librerías para el cliente mysql. Para ello se deben instalar los paquetes libmysqlclient15off y libmysqlclient15-dev.
- En la configuración del PHP se debe especificar el directorio /usr/ donde se encuentran las nuevas librerías instaladas. Para ello se agrega la opción --with-mysql=/usr/

Otros aspectos de la arquitectura del cluster

El cluster cuenta con un controlador en el cual se manejan dos bases de datos virtuales, cada una de estas bases de datos virtuales mantiene un *pool* de conexiones de tamaño variable con cada *backend*. La conexión entre Myosotis y Sequoia se realiza utilizando el driver JDBC proporcionado por Sequoia; las aplicaciones clientes establecen las conexiones con Myosotis de la misma manera que lo harían

con un servidor MySQL pero utilizando el puerto 9999. En la figura 3.13 se puede apreciar con más detalle las características y la arquitectura de la solución propuesta.

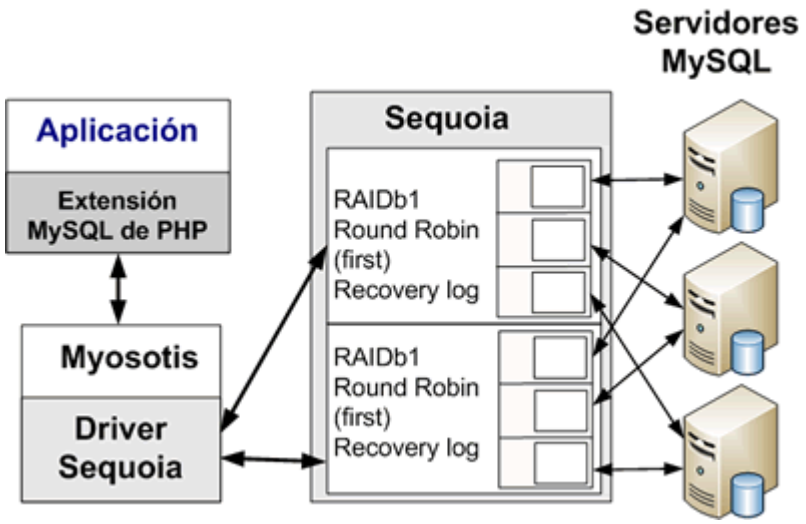

Figura 3.13. Arquitectura del cluster Myosotis y Sequoia (primera variante).

Para el balanceo de carga se utilizó un algoritmo Round Robin y la replicación de los datos se realizó de forma completa en cada servidor utilizando el esquema de distribución de datos RAIDb1, se empleó además una política "first" para el envío de respuestas a las consultas de modificación efectuadas en los servidores. La selección del algoritmo de balanceo de carga y el método de replicación se realizó siguiendo los mismos criterios mencionados en la solución anterior.

El cluster cuenta con un mecanismo de recuperación ante fallos proporcionado por la base de datos de recuperación (*recovery log*) de Sequoia. Se decidió utilizar una base de datos MySQL 4.0.1 en el nodo balanceador donde se almacenan toda la información necesaria para la recuperación. Se utilizó la versión 4.0.1 para ser consistente con las versiones de MySQL utilizadas por la aplicación, sin embargo como base de datos de recuperación puede utilizarse cualquier versión de MySQL u otro de los gestores de bases de datos soportados por Sequoia.

La configuración de Myosotis es muy sencilla, sólo se debe indicar a Myosotis donde localizar el driver Sequoia y establecer algunos parámetros relacionados con la conexión tales como: un usuario, una contraseña, la dirección ip del controlador y las bases de datos virtuales a las que se puede acceder. Para más detalle de cómo realizar la configuración ver documento adjunto *"Instalación y configuración de la solución Myosotis y Sequoia para RUS.pdf"*.

3.2.4. Diseño de la solución Myosotis y Sequoia para el tercer grupo de aplicaciones

Para las aplicaciones del tercer grupo se propone una variante de la solución de cluster anterior que integra el uso del balanceador de carga Sequoia y Myosotis, este último actúa como intermediario entre la aplicación y el balanceador para que se pueda establecer una conexión entre ambos.

El cluster se compone de un grupo de servidores de bases de datos MySQL 5.0.3 desplegados en varios nodos. Se utiliza además un nodo que juega el papel de controlador Sequoia desde donde se realiza la replicación y el balanceo de carga, en este nodo se incluye además el *software* Myosotis y de esta manera se logra una interacción entre la aplicación y el cluster como se muestra en la figura 3.14.

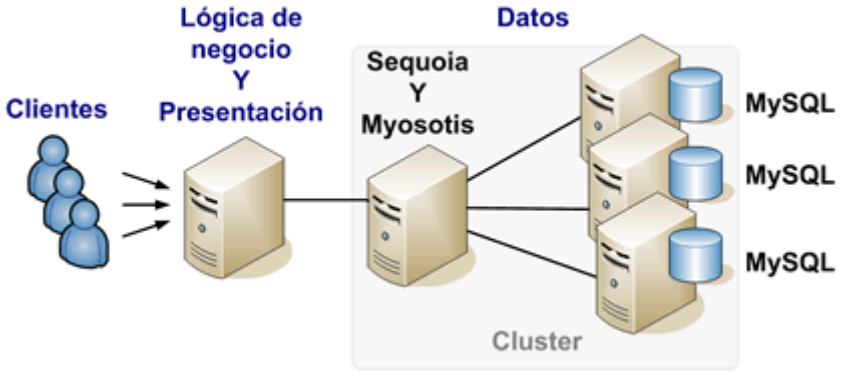

Figura 3.14. Propuesta de diseño del cluster Sequoia y Myosotis (segunda variante).

La instalación del servidor de lógica de negocio, a su vez servidor de capa de presentación, no requiere modificaciones ya que el acceso al cluster a través de las librerías utilizadas se realiza de manera satisfactoria.

La arquitectura del cluster comprende un controlador Sequoia el cual maneja una única base de datos virtual. En el controlador se debe definir un esquema para cada *backend* manejado por la base de datos virtual, esta definición permite al cluster realizar llamados a los procedimientos almacenados en cada servidor de bases de datos. El uso de procedimientos almacenados es una característica que diferencia a este grupo de aplicaciones de las anteriores. Esta característica se tuvo en cuenta para la realización del diseño de esta solución.

El esquema de la base de datos se configuró para que se construyera de manera dinámica tomando la información necesaria de cada *backend*. Se debe indicar al controlador como manejar las llamadas a los procedimientos almacenados, para ello se definió una regla por defecto para cada *backend*. Esta configuración se realiza en el fichero de configuración perteneciente a la base de datos virtual, ver documento adjunto "*Instalación y configuración de la solución Myosotis y Sequoia para BM.pdf".*

Se utilizó un algoritmo de balanceo de carga Round Robin y un método de replicación RAIDb1 con política “first” por las mismas razones descritas en soluciones anteriores.

La arquitectura del cluster para esta variante se puede apreciar con más de talle en la figura 3.15.

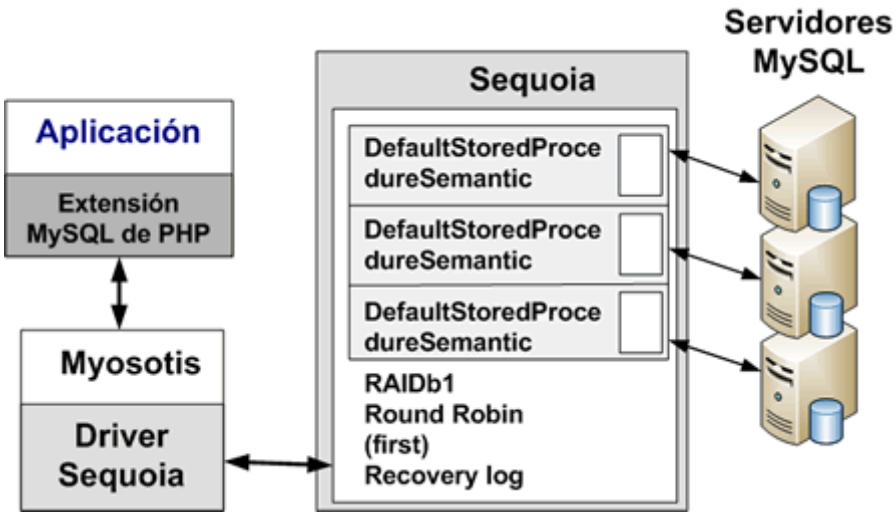

Figura 3.15. Arquitectura del cluster Myosotis y Sequoia (segunda variante).

3.3. Consideraciones para el despliegue de las aplicaciones

El diseño de las soluciones descritas incluye además una serie de consideraciones de importancia a tener en cuenta a la hora de realizar el despliegue de las mismas y para garantizar el buen funcionamiento de cada solución propuesta. Estas consideraciones incluyen un reajuste de la red donde se desplegarán las soluciones, un grupo de acciones a tener en cuenta para el monitoreo así como el cumplimiento de un grupo de medidas de seguridad. Cada uno de estos elementos se describe con más detalle en los siguientes epígrafes.

Propuesta de reajuste del diseño de la red

El acceso a las bases de datos y la gestión de los sistemas, que se encuentran ubicados en el nodo central de Infomed, se realizan desde redes externas que pertenecen a la red privada del MINSAP 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 los despliegues de las aplicaciones comienzan a incluir *cluster* de balanceo de carga para servidores de bases de datos, el número de servidores involucrados aumenta y con ello el número de direcciones IP reales que serían necesarias. Debido a la escasez de estas últimas esto sería un proceso insostenible.

Para solucionar este problema se propone reajustar el diseño de la red para ubicar a todos los servidores en una subred IP privada.

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 facilitar estos procesos.

Propuestas de modificaciones a la gestión del monitoreo

El monitoreo de las aplicaciones desplegadas en la red de Infomed se realiza utilizando las herramientas Nagios y MRTG (*Multi Routing Traffic Grapher*). La primera se utiliza en la detección de fallas en la red, los servidores y los servicios y la segunda para conocer el consumo de los recursos en cada servidor de manera histórica.

Cuando se efectúe el despliegue de las soluciones propuestas se deberá incluir cada uno de los servidores que conforman el cluster al monitoreo realizado para servidores anteriormente desplegados en la red de Infomed. El monitoreo se hará de la misma manera que se describe anteriormente y éste puede dividirse en dos partes: monitoreo del rendimiento del cluster y monitoreo del consumo de recursos en los servidores.

Monitoreo del rendimiento del cluster

El rendimiento del cluster se medirá utilizando como indicador los tiempos de respuesta alcanzados a las peticiones realizadas al mismo. Para hacer la medición se diseñará un script que realice una consulta al balanceador de carga y genere cierto nivel de procesamiento.

A través de la herramienta Nagios se realizará el chequeo sistemático del cluster mediante la ejecución del script y se medirá el tiempo en que demora en ejecutarse la consulta. Para que se produzca una alarma se fijará un tiempo de respuesta crítico. Las alarmas se generan bajo tres niveles de estado, un nivel de estado aceptable, un nivel de estado de alerta y un nivel de estado de falla. Una vez alcanzado uno de los niveles de alerta o de falla el administrador del sistema recibirá una notificación que puede ser un mensaje de alarma proporcionado por la herramienta Nagios.

El estado de alerta puede ser definido para que se genere una alarma cuando los tiempos de respuesta difieren en un 5% del tiempo crítico, mientras la diferencia sea mayor se considerará como

estado aceptable y si se alcanza el tiempo crítico se informará como falla. Para definir cuándo se debe producir una alarma, así como el tiempo crítico de respuesta se debe realizar un estudio detallado del comportamiento del sistema y de los tiempos de respuesta promedios obtenidos cuando se realizar varias consultas, como la definida en el script, en un ambiente de producción real. Por tal motivo estos criterios pueden variar a consideración del administrador del sistema.

Monitoreo del consumo de recursos en los servidores

El monitoreo para detectar el consumo de los recursos en los servidores se realiza a través de la herramienta MRTG y Nagios. La primera permite realizar un análisis retrospectivo de cómo se han utilizado los recursos en un período de tiempo determinado, con la segunda se pueden monitorear los recursos de la misma forma a como se hizo para determinar el rendimiento del cluster.

Para analizar el consumo de recursos con la herramienta Nagios y programar el sistema de alarmas se define un límite o nivel de umbral para cada servidor en el consumo de recursos de memoria RAM y CPU. Estos recursos son los más utilizados por los servidores de bases de datos por lo que representan un punto crítico de cada servidor. También se definirán varios estados de alarma (aceptable, alerta y falla) de la misma forma que se definió en el epígrafe anterior.

Par definir los niveles críticos en el consumo de recursos de los servidores de bases de datos se debe realizar un estudio del por ciento utilizado históricamente de sus recursos (memoria RAM, tráfico de red, CPU, etc) bajo un entorno de producción en tiempo real. Los valores históricos de utilización de estos recursos son proporcionados por la herramienta MRTG.

El monitoreo de los recursos debe realizarse a través de un script el cual será ejecutado periódicamente desde la herramienta Nagios para chequear el por ciento de recursos utilizado en cada uno de los servidores de bases de datos.

Propuestas de modificaciones a la gestión de la seguridad

Se propone el uso de dos niveles de seguridad, el primero lo constituye el cortafuegos o enrutador que realiza el filtraje de paquetes IP a las computadoras que conforman el cluster desde el exterior, este cortafuegos o enrutador debe tener como regla por defecto “denegar todo” y solo permitir el acceso imprescindible.

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 prestan haciendo uso de reglas *IPTABLES.* Para implementar estas políticas de seguridad, sin que se afecte la comunicación entre los servidores, en necesario que se permitan las siguientes conexiones:

Variante 1: LVS (*Linux Virtual Server*) y replicación MySQL.

- Conexiones entre los balanceadores de carga (primario y de respaldo). Ambos servidores deben permitir el acceso mutuo a través del puerto 3486 del protocolo UDP.
- Conexiones desde los balanceadores de carga a los servidores de bases de datos. Los servidores de bases de datos deben permitir el acceso a través del puerto 3306 del protocolo TCP.
- Conexiones desde los servidores de bases de datos (esclavos) al *master* de la replicación. El servidor *master* debe permitir el acceso a los servidores de bases de datos a través del puerto 3306 del protocolo TCP.
- El balanceador de carga y el *master* de la replicación deben permitir el acceso a través del puerto 3306 del protocolo TCP para uso de los clientes (servidores de lógica de negocio o servidores de capa de presentación y lógica de negocio).
- El cortafuegos o enrutador debe permitir el acceso desde las computadoras de administración y monitoreo a través de los puertos 22, 3306, y los puertos necesarios para utilización de las herramientas de monitoreo Nagios y MRTG, todos estos del protocolo TCP.

Variante 2: Sequoia y Myosotis.

- Conexiones desde el balanceador de carga a los servidores de bases de datos. Los servidores de bases de datos deben permitir el acceso a través del puerto 3306 del protocolo TCP.
- El balanceador de carga debe permitir el acceso a través del puerto 9999 del protocolo TCP para uso de los clientes (servidores de lógica de negocio o servidores de capa de presentación y lógica de negocio).
- El cortafuegos o enrutador debe permitir el acceso desde las computadoras de administración y monitorio a través de los puertos 22, 3306, y los puertos necesarios para utilización de las herramientas de monitoreo Nagios y MRTG, todos estos del protocolo TCP.

3.4. Conclusiones

El diseño de las soluciones se realizó teniendo en cuenta las características del grupo de aplicaciones para el cual se propone el cluster. De esta forma se confeccionó una propuesta de diseño para cada uno de los grupos de aplicaciones y en el caso del primer grupo, representado por la aplicación Registro de Unidades de Salud (RUS), se propusieron dos alternativas de cluster: LVS y replicación MySQL y el cluster con Myosotis y Sequoia.

Cada uno de los diseños propuestos será desplegado en un escenario de pruebas en el laboratorio y se le aplicarán un grupo de pruebas para su validación. El diseño de cada solución incluyó además algunas consideraciones para el despliegue del cluster en su entorno real así como requerimientos de seguridad y una propuesta de cómo se debe realizar el monitoreo del cluster.

CAPÍTULO 4: PRUEBAS PARA VALIDAR LAS SOLUCIONES Y ANÁLISIS DE RESULTADOS

4.1. Descripción del proceso de pruebas

El proceso de pruebas se define como:

"*Una actividad en la cual un sistema o componente es ejecutado bajo condiciones específicas, se observan o almacenan los resultados y se realiza una evaluación de algún aspecto del sistema o componente*". (Pressman 2001)

Con el propósito de garantizar el correcto funcionamiento de cualquier sistema es necesario realizar una etapa de prueba donde se demuestre mediante los distintos tipos de pruebas existentes que dicho sistema está listo para su despliegue.

Estas pruebas se realizan fundamentalmente con los siguientes objetivos:

- Proveer confianza en el sistema.
- Identificar las áreas de debilidad (áreas donde el sistema esté más propenso a fallos debido a una serie de condiciones que se pueden presentar una vez que el sistema esté desplegado como por ejemplo: carga de trabajo excesiva, agotamiento de recursos en los servidores, los distintos tipos de ataques que puede recibir, etc.)
- Establecer un grado de calidad del sistema.
- Proveer un entendimiento general de cómo funciona el mismo.
- Probar además que usable y operable.
- Para obtener suficiente información que permita tomar una decisión objetiva en cuanto al despliegue. (Meier et al. 2007)

Para el caso particular de esta investigación con el objetivo de comprobar el correcto funcionamiento de los sistemas de clusters de bases de datos y medir su rendimiento se decidió realizar un conjunto de pruebas las cuáles se clasifican fundamentalmente en tres tipos:

1 - Pruebas de configuración:

Estas pruebas tienen como principal objetivo el de comprobar el correcto funcionamiento de la aplicaciones sobre una configuración diferente de *software* o *hardware* que la utilizada en el despliegue original de la aplicación.

Con la realización de las pruebas de configuración se obtienen los siguientes beneficios:

- Ayudan demostrar que el sistema funciona correctamente sobre la nueva configuración de *hardware* o *software* que se está probando (la solución de cluster), prestando especial atención a las funcionalidades críticas y casos de uso representativos de la aplicación.
- Ayudan a validar la solución de cluster.(Meier et al. 2007)

2 - Pruebas de carga:

Las pruebas de carga son un tipo de prueba de rendimiento utilizadas para validar y evaluar la aceptabilidad de los límites operacionales de un sistema bajo distintos volúmenes de trabajo mientras la solución de cluster se mantiene constante (la misma cantidad de nodos). Y en otras variantes, el volumen de trabajo se mantiene constante y la configuración de la solución de cluster es variada (aumentando la cantidad de nodos).

Las pruebas de carga son comúnmente realizadas para lograr uno de los siguientes objetivos:

- Evaluar los resultados obtenidos contra los criterios de rendimiento.
- Encontrar la(s) fuente(s) de los problemas de rendimiento.
- Buscar niveles de rendimiento.

En el caso particular de la realización de pruebas a los sistemas de cluster de bases de datos se han establecido distintas métricas a recolectar como son: el tiempo de respuesta promedio de las consultas, cantidad de transacciones por unidad de tiempo, por ciento de uso de CPU *(Central Processing Unit /* Unidad Central de Procesamiento*)*, por ciento de uso de memoria RAM *(Random Access Memory /* Memoria de Acceso Aleatorio*)*, y tráfico de red pero prestando mayor atención y tomando como métrica principal el tiempo de respuesta promedio de las consultas.

Con la realización de las pruebas de cargas se obtienen los siguientes beneficios:

- Ayudan a determinar cuanta carga el *hardware* puede soportar antes de que los límites de utilización de recursos (por ciento de uso de CPU, por ciento de uso de memoria RAM y tráfico de red) sean excedidos.
- Ayudan a recopilar datos de vital importancia para la planificación de la escalabilidad y capacidad del sistema.
- Ayudan a evaluar la idoneidad del balanceador de carga.

En el caso de esta investigación se realizarán fundamentalmente dos variantes de pruebas de carga. La primera se realizará manteniendo un nivel de concurrencia fijo con el objetivo de analizar como se mejoran los tiempos de respuesta promedio de la aplicación desplegada sobre el cluster a medida que se vayan aumentando la cantidad de nodos del mismo. Y la segunda se realizará manteniendo el número de nodos fijo con el objetivo de analizar como se deterioran los tiempos de respuesta promedio a medida que se vayan aumentando los niveles de concurrencia.

Tener en cuenta que estas pruebas se realizarán sobre computadoras personales con diferente nivel de recursos que los servidores reales en los que serán desplegadas las aplicaciones. Esto trae consigo que los resultados obtenidos no sean exactos pero ayudan a entender como mejora o se deteriora el funcionamiento de las aplicaciones desplegadas sobre el cluster.

3 - Pruebas de *benchmark*:

Estas pruebas realizan para comparar el comportamiento de los sistemas o las aplicaciones sobre diferentes configuraciones de *software* y *hardware*. La realización de este tipo de prueba consiste en medir el rendimiento de las aplicaciones sobre determinada configuración de *software* o *hardware* y comparar los resultados obtenidos con configuraciones similares.

Con la realización de las pruebas de *benchmark* se persigue como objetivo principal, dadas dos o más configuraciones de *software* y *hardware* que soporten una misma aplicación, el de encontrar cuál sería el entorno idóneo para el despliegue de dicha aplicación.

Procedimiento general

En la figura 4.1 se muestra los pasos a seguir de manera general para la realización de las pruebas.

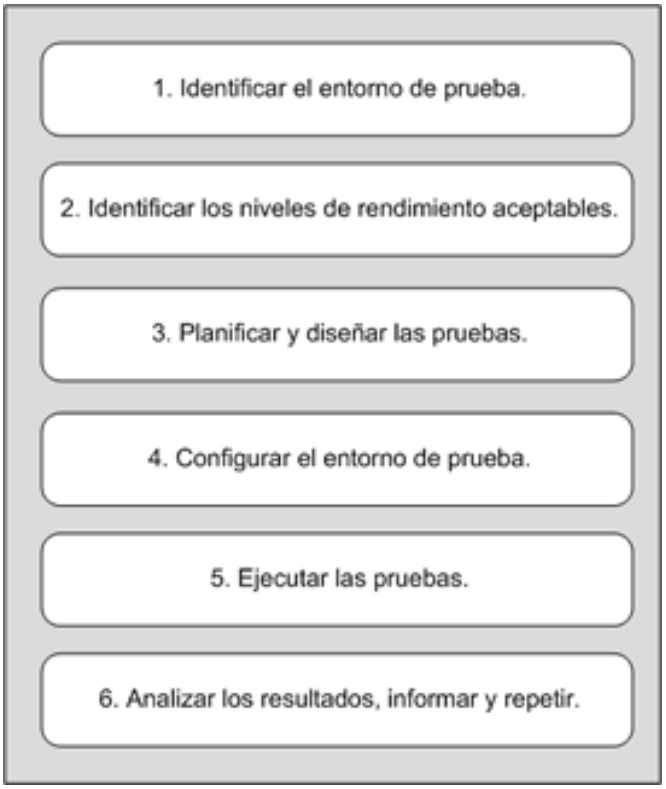

Figura 4.1. Procedimiento general de pruebas.

1. *Identificar el entorno de prueba:* Identificar el entorno físico de prueba y el entorno de producción, así como las herramientas y recursos disponibles para la realización de las mismas. El entorno físico incluye *hardware*, *software* y configuraciones de red. Tener un conocimiento profundo del entorno de prueba permite, en primer lugar, un diseño y planificación de la prueba más eficiente.
2. *Identificar los niveles de rendimiento aceptables:* Identificar el tiempo de respuesta, rendimiento y, los objetivos y limitaciones de la utilización de recursos.
3. *Planificar y diseñar las pruebas:* Identificar los principales escenarios, definir los datos de prueba y establecer las métricas que deben recolectarse.
4. *Configurar el entorno de prueba:* Preparar el entorno de prueba, herramientas y recursos necesarios para ejecutar cada una de las estrategias a medida que cada componente necesario para la prueba este disponible. Asegurarse de que el entorno de prueba se ha instrumentado para el monitoreo de recursos según sea necesario.

5. *Ejecutar la prueba:* Ejecutar y monitorear las pruebas. Validar las pruebas, los datos de prueba y los resultados recolectados. Ejecutar las pruebas validadas para su análisis mientras se este monitoreando la prueba y el entorno de prueba.
6. *Analizar los resultados, informar y repetir la prueba:* Consolidar y compartir los resultados. Cuando todas las métricas estén dentro de los límites aceptados, ninguno de los umbrales establecidos haya sido violado y toda la información necesaria haya sido recolectada, entonces se ha terminado de probar esa configuración en particular, en caso contrario, rediseñar y repetir la prueba o declararla fallida.

Es válido aclarar que para la realización de cada tipo pruebas no hay que ejecutar todos los pasos mencionados anteriormente. Este procedimiento se realizó con el objetivo de brindar una idea general del proceso pero puede estar sujeto a cambios para cada una de las pruebas.

4.2. Pruebas a la propuesta de solución LVS y Replicación MySQL para RUS

4.2.1. Diseño de pruebas de configuración

Propósito de las pruebas

Comprobar que la propuesta de solución de cluster de bases de datos diseñada con LVS (*Linux Virtual Server*) y replicación MySQL para la aplicación Registro de Unidades de Salud realiza todas las funcionalidades a probar correctamente.

Descripción del despliegue

Cluster de balanceo de carga para bases de datos que está compuesto por dos balanceadores de carga, un servidor *master* de la replicación y dos servidores de bases de datos. El cluster incluye funcionalidades de balanceo de carga para las consultas de lectura (select) y replicación para las consultas de modificación de las base de datos (insert, update, delete), así como, de recuperación ante fallos en los servidores de bases de datos y alta disponibilidad para el servicio de balanceo de carga.

Funcionalidades del sistema a probar

1. Balanceo de carga.
2. Replicación.
3. Alta disponibilidad en el balanceador de carga.
4. Recuperación ante fallos de los servidores de bases de datos.
5. Correcto funcionamiento de la aplicación.

Comportamiento esperado de las funcionalidades del sistema a probar

1. Balanceo de carga: Se espera que al llegar las consultas de lectura (select) al balanceador de carga este las distribuya entre los servidores de bases de datos de la forma más equitativa posible.
2. Replicación: Se espera que cuando se realice alguna operación de modificación sobre el *master,* la misma se realice además, sobre todos los servidores de bases de datos mediante el proceso de replicación.
3. Alta disponibilidad en el balanceador de carga: Se espera que si el balanceador de carga primario deja de prestar sus servicios el balanceador de carga de respaldo sea capaz de recuperar el sistema automáticamente y sin la intervención del administrador.
4. Recuperación ante fallos de los servidores de bases de datos: Se espera que si uno de los servidores de bases de datos deja de prestar sus servicios el balanceador de carga lo detecte y deje de reenviarle consultas para que esto no afecte el funcionamiento del sistema.
5. Funcionamiento correcto de la aplicación: Se espera que el comportamiento de la aplicación al ejecutarse sobre la propuesta de solución diseñada para ella y sobre el despliegue original sea el mismo. Es decir, que la existencia del cluster sea transparente al usuario.

Planificación de la ejecución de las pruebas

1. Para analizar como el balanceador de carga distribuye las conexiones entre los servidores de bases de datos se va a realizar una navegación por la aplicación desde las computadoras clientes y en los servidores de bases de datos se va a analizar la cantidad de consultas que se realizaron sobre cada uno de ellos, esta información se va a extraer de los *logs* de acceso de los servidores MySQL.
2. Para analizar como se comporta el proceso de replicación se va a realizar una navegación por la aplicación desde las computadoras clientes, específicamente realizando operaciones de modificación, y posteriormente se va a analizar si las consultas que se realizaron sobre el *master,* se realizaron a su vez, sobre todos los servidores de bases de datos, esta información se va a extraer de los *logs* de acceso de los servidores MySQL.
3. Para probar la alta disponibilidad se va a realizar una navegación por la aplicación desde las computadoras clientes y se le va a retirar el servicio de red al balanceador de carga primario para comprobar que el sistema continúe brindando sus servicios.
4. Para probar la recuperación ante fallos de los servidores de bases de datos se va realizar una navegación en la aplicación desde las computadoras clientes y se le va a retirar el servicio de red

uno de los servidores de bases de datos para comprobar que el sistema continúe brindando sus servicios.

5. La aplicación Registro de Unidades de Salud se desplegará en dos niveles, uno para la capa de presentación y lógica de negocio y otro para la capa de datos. De manera simultánea se realiza otro despliegue sobre el cluster de bases de datos. La prueba consiste en navegar en las aplicaciones de ambos despliegues y comparar el comportamiento de cada una prestando especial atención a los casos de uso representativos de la misma.

4.2.2. Recogida de datos y análisis de resultados de las pruebas de configuración

Durante la primera iteración de pruebas de configuración realizadas a la aplicación RUS (Registro de Unidades de Salud) en cuatro de las cinco funcionalidades a probar se obtuvieron resultados positivos. Cuando se comprobó el balanceo de carga los resultados demostraron que no existía, por parte de la aplicación, la separación en las consultas de lectura y escritura que es uno de los requisitos no funcionales expresados en el documento de arquitectura de Softel “Arquitectura, normas y tecnologías para el desarrollo de aplicaciones informáticas para la Salud Pública en Cuba”.

Debido a que el código de la aplicación con que se estaban realizando las pruebas estaba encriptado, una vez analizados estos resultados, se procedió a la gestión de obtener el código claro de dicha aplicación con el objetivo de reprogramarla y lograr esta separación de las consultas de lectura y escritura. Después de cumplida esta tarea se realizó la segunda iteración en las pruebas de configuración donde todos los resultados fueron positivos.

4.2.3. Diseño de pruebas de carga

Propósitos de las pruebas

1. Encontrar el número de usuarios concurrentes, que al navegar por la aplicación, provocan que el servidor de bases de datos, sobre el que esta se ejecuta, llegue al límite en el consumo de alguno de sus recursos.
2. Demostrar como al aumentar el número de servidores de bases de datos en el cluster de balanceo de carga, manteniendo el nivel de concurrencia constante, disminuye el tiempo de respuesta del sistema y el consumo de recursos en los servidores de bases de datos.
3. Demostrar como al aumentar el nivel de concurrencia en el cluster de balanceo de carga, manteniendo la cantidad de servidores de bases de datos constante, aumenta el tiempo de respuesta del sistema y el consumo de recursos en los servidores de bases de datos.

Descripción de los despliegues

1. La aplicación estará desplegada sobre dos servidores uno para la aplicación (capa de presentación y lógica de negocios) y otro para la base de datos (capa de datos).
2. La aplicación tendrá su capa de datos desplegada sobre un cluster de balanceo de carga. Este estará compuesto por un balanceador de carga, un *master* de la replicación y el número de servidores de aplicación necesarios para la realización de la prueba. La capa de presentación y la lógica de negocio estarán sobre servidores externos al cluster.

Planificación de la ejecución de las pruebas

Desde tres computadoras diferentes se va a utilizar la herramienta JMeter para generar un nivel de concurrencia de usuarios navegando por la aplicación.

Después de un período de calentamiento de varios minutos para que la aplicación estabilice su funcionamiento, se realizará la medición para obtener el tiempo promedio que tarda el sistema en devolver todas las consultas solicitadas.

Para realizar la medición también se va a utilizar la herramienta JMeter, pero desde una cuarta computadora. La herramienta se va a configurar para que realice la navegación de un usuario un total de 10 veces y además se le va a añadir un elemento al JMeter para la recogida del tiempo de respuesta del sistema para cada consulta que se le solicite. Este elemento se agrega porque, para los resultados de la prueba, es importante analizar los tiempos de respuesta promedio de un conjunto de páginas y no los tiempos de respuesta individuales de cada consulta.

Simultánea a la medición del tiempo de respuesta se realizará, utilizando la herramienta Sysstat, la medición del consumo de recursos en los servidores de bases de datos, en el balanceador de carga y en el *master* de la replicación. La herramienta se ejecuta, a través de un script, con los parámetros necesarios para que cada dos segundos realice una medición hasta completar un total de 200.

Con el objetivo de encontrar el número de usuarios concurrentes que soporta un servidor de bases de datos en el despliegue original de la aplicación se comenzará la prueba con cinco usuarios y se va a repetir la prueba hasta encontrar el límite del servidor, añadiendo cinco usuarios concurrentes más en cada iteración de la misma.

Además, para poder mostrar como disminuyen los tiempos de respuesta promedio del sistema y el consumo de recursos en los servidores de bases de datos, la prueba será realizada variando la cantidad de nodos del cluster de balanceo de carga desde uno hasta seis servidores de bases de datos, manteniendo el mismo nivel de concurrencia.

Para poder mostrar como aumentan los tiempos de respuesta promedio del sistema y el consumo de recursos en los servidores de bases de datos, la prueba será realizada manteniendo la misma configuración de cluster (cluster de balanceo de carga con 3 servidores de bases de datos) con seis niveles de concurrencia diferentes.

4.2.4. Recogida de datos y análisis de resultados de las pruebas de carga

Datos recogidos

Para demostrar la disminución de los tiempos de respuesta del sistema y del consumo de recursos en los servidores de bases de datos con el aumento de los nodos del cluster se recogieron los siguientes datos: tiempos de respuesta promedio, consumo de recursos promedio en los servidores de bases de datos y el consumo de recursos promedio en el balanceador de carga.

En la tabla 4.1 se muestran los tiempos de respuesta promedio del sistema.

Tabla 4.1. Tiempos de respuesta promedio del sistema.

Número de servidores de bases de datos fuera y dentro del cluster (c)	Tiempo de respuesta promedio (s)
1	1.592
1 (c)	1.664
2 (c)	0.803
3 (c)	0.545
4 (c)	0.540
5 (c)	0.530
6 (c)	0.520

En la tabla 4.2 se muestra un resumen de los valores promedio en el consumo de recursos en los servidores de bases de datos.

Tabla 4.2. Consumo de recursos promedio en los servidores de bases de datos.

Recursos	Número de servidores de bases de datos fuera y dentro del cluster (c)						
	1	1 (c)	2 (c)	3 (c)	4 (c)	5 (c)	6 (c)
% de uso de CPU	99.02	99.67	96.44	48.71	47.98	38.32	32.01
% de consumo de memoria RAM	12.50	14.28	13.16	12.01	10.52	9.18	8.52
% de uso de la red	0.61	5.16	4.34	3.84	3.55	2.70	2.11

En la tabla 4.3 se recogen los valores promedios del consumo de recursos en el balanceador de carga.

Tabla 4.3. Consumo de recursos promedio en el balanceador de carga.

Recursos	**Número de servidores de bases de datos dentro del cluster (c)**					
	1 (c)	2 (c)	3 (c)	4 (c)	5 (c)	6 (c)
% de uso de CPU	0.49	0.56	0.62	0.73	0.73	0.75
% de consumo de memoria RAM	21.16	20.96	20.89	20.13	19.76	18.80
% de uso de la red	0.28	0.53	0.64	0.92	0.92	0.93

Análisis de los resultados

Como se puede apreciar en el gráfico de la figura 4.2 con el aumento del número de servidores de bases de datos en el cluster de balanceo de carga para la capa de datos de la aplicación Registro de Unidades de Salud los tiempos de respuesta promedio disminuyen.

Al realizar la comparación entre los tiempos de respuesta promedio de la aplicación sobre su despliegue original y los de la aplicación desplegada en el cluster con un solo nodo se pudo apreciar que los tiempos fueron menores en el primer caso. Esto se debe a que en el segundo caso existen pequeñas demoras introducidas por el balanceador de carga que retrasan la respuesta de las consultas. Sin embargo, ya con dos servidores reales en la configuración del cluster los tiempos de respuesta son menores y continúan disminuyendo con el aumento de los nodos.

En la gráfica el punto aislado representa el servidor real utilizado para medir cuantos carga soportaba la aplicación en su despliegue original y la función representa las distintas configuraciones del cluster. Dicha función permite concluir que para la carga que se estaba generando durante la realización de esta prueba, con tres nodos en el cluster es suficiente.

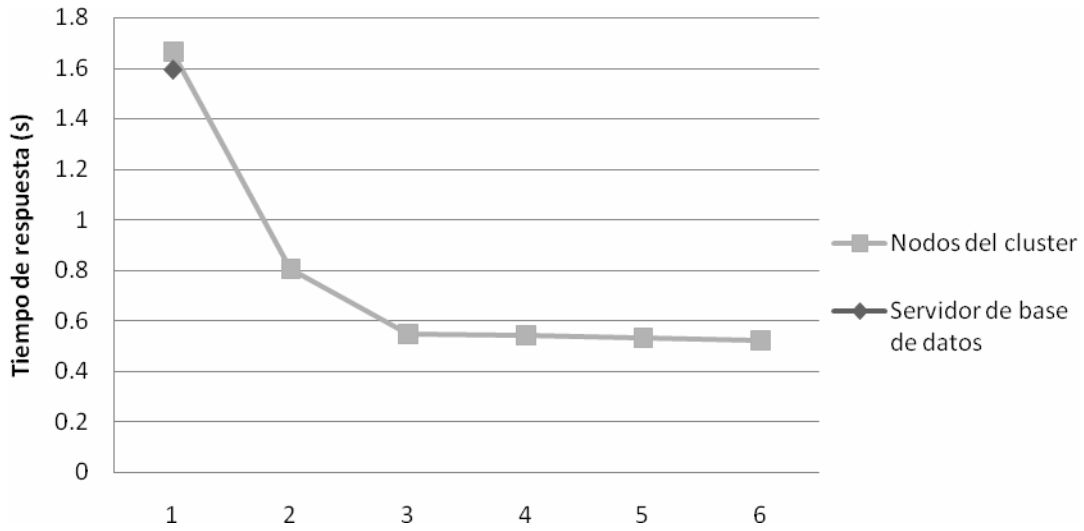

Figura 4.2. Tiempos de respuesta promedio contra número de nodos.

De manera semejante a lo que pasa con los tiempos de respuesta, con el aumento del número de servidores de bases de datos en el cluster, el consumo de recursos en estos también disminuye, debido a que cada servidor tiene que atender cada vez menos consultas. Esto se puede observar en las figuras 4.3 y 4.4.

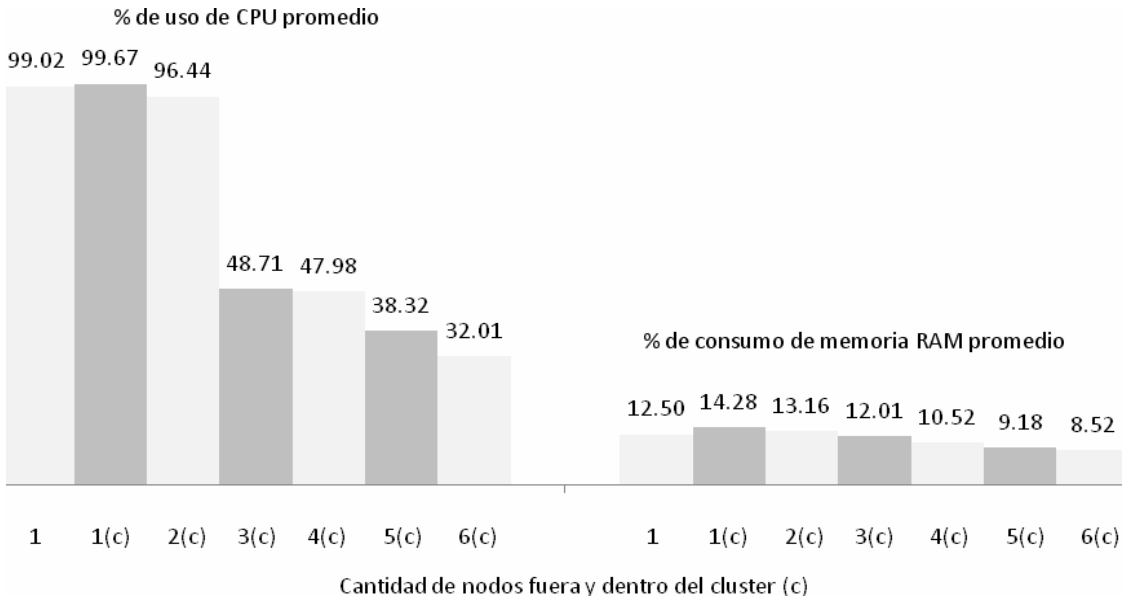

Figura 4.3. Consumo de recursos promedio (CPU y RAM) en los servidores de bases de datos.

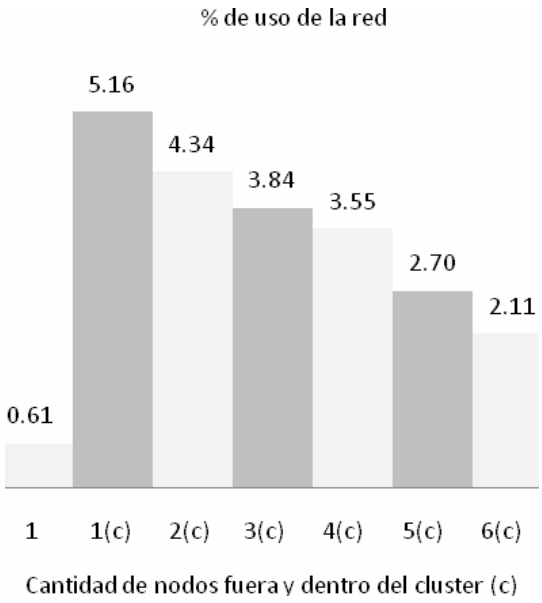

Figura 4.4. Consumo de recursos promedio (tráfico de red) en los servidores de bases de datos.

Con el aumento del número de servidores de bases de datos el consumo de CPU y el tráfico de red en el balanceador de carga aumentan, pero para la prueba realizada no alcanzan niveles preocupantes, como se muestra en las figuras 4.5 y 4.6.

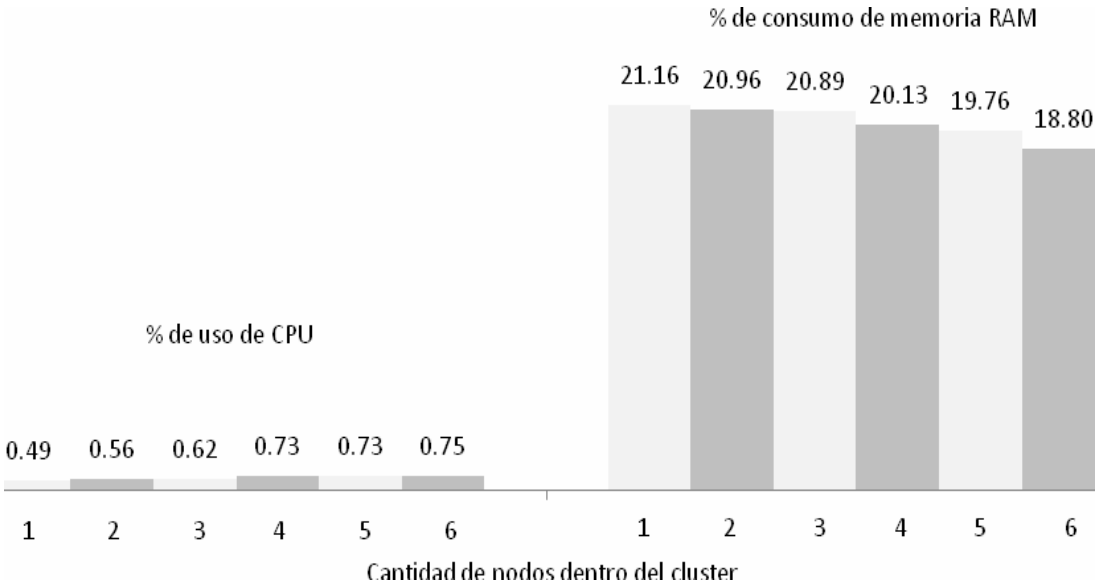

Figura 4.5. Consumo de recursos (CPU, RAM) promedio en el balanceador de carga.

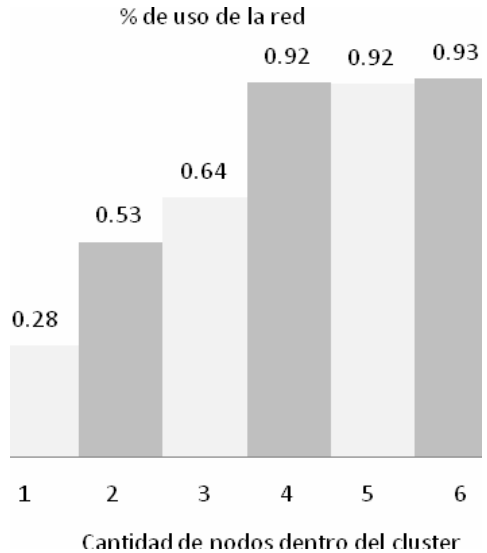

Figura 4.6. Consumo de recursos (tráfico de red) promedio en el balanceador de carga.

De manera semejante, con el aumento del número de servidores de bases de datos el consumo de CPU y el tráfico de red en el *master* de la replicación aumentan, pero para la prueba realizada tampoco alcanzan niveles preocupantes, como se muestra en las figuras 4.7 y 4.8.

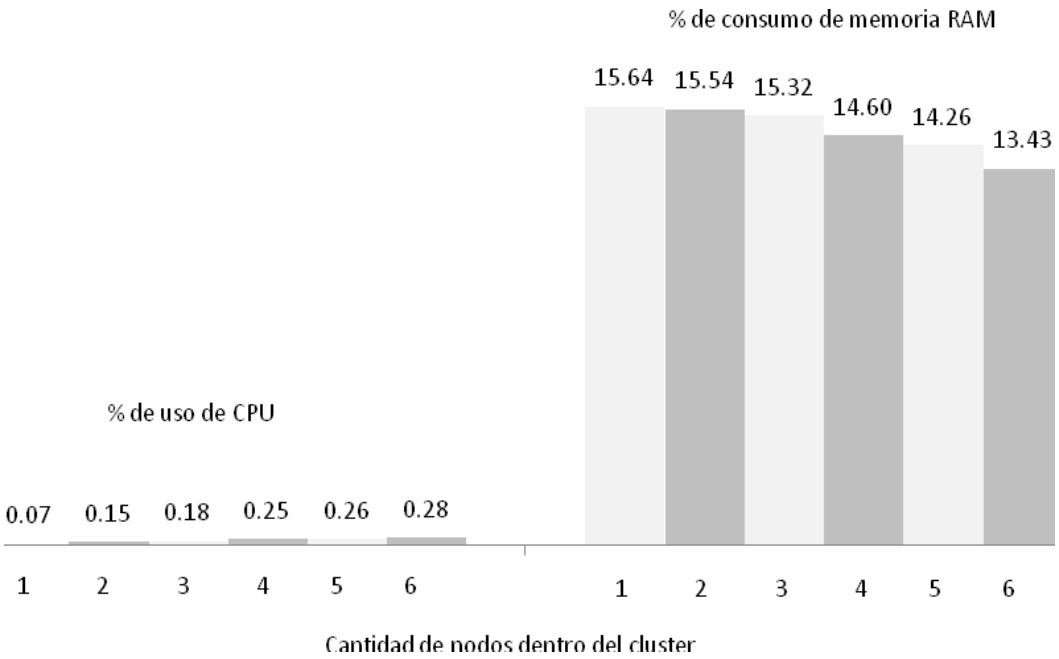

Figura 4.7. Consumo de recursos (CPU, RAM) promedio en el *master* de la replicación.

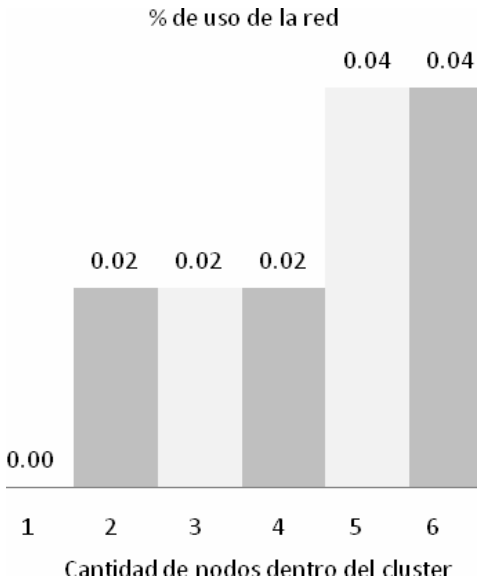

Figura 4.8. Consumo de recursos (red) promedio en el *master* de la replicación.

Lo más importante de analizar el consumo de recursos en el balanceador de carga y en el *master* de la replicación es estimar el límite en el escalamiento del cluster, para así, tener una idea del número máximo de servidores de bases de datos que puede manejar el sistema, el cual estaría limitado por el consumo de recursos en uno de estos dos servidores.

Para hacer ese análisis se buscaron las líneas de tendencia para el consumo de cada de uno de los recursos que se están monitoreando y al graficarlas se obtiene el resultado que se muestra en la figura 4.9. Como se puede apreciar en el gráfico se excluyó del análisis el por ciento de uso de memoria RAM debido a que en ambos servidores el consumo de este recurso disminuye por lo que no sería una limitante en el escalamiento del cluster.

Además, es válido aclarar que, como se puede observar en la figura, el comportamiento de las curvas es muy accidentado. Debido a esto, la función que se obtuvo como resultado del ajuste de curvas que se realizó, utilizando el método lineal, no describe el comportamiento del consumo de los recursos e introduce un error de aproximación demasiado grande. Para lograr una función más acertada sería necesario continuar realizando el experimento para obtener más puntos en la gráfica, por lo que las funciones que se muestran en gráfico no son válidas desde un punto de vista matemático de rigor y su único objetivo es mostrar la tendencia que tiene el consumo de recursos en ambos servidores (balanceador de carga y *master* de la replicación).

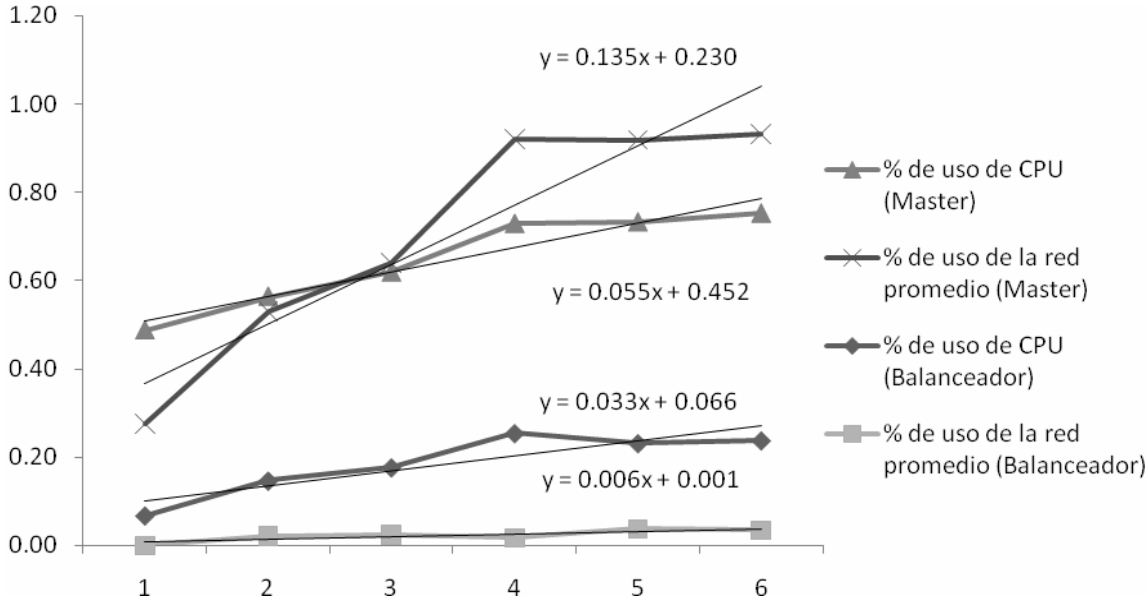

Figura 4.9. Líneas de tendencia del consumo de recursos para el balanceador y el *master* de la replicación.

De manera homóloga se realizará un análisis para demostrar como para una misma configuración de cluster al aumentar los niveles de concurrencia, aumentarán los tiempos de respuesta del sistema y el consumo de recursos en los servidores de bases de datos.

Datos recogidos

En la tabla 4.4 se muestran los tiempos de respuesta promedio del sistema.

Tabla 4.4. Tiempos de respuesta promedio del sistema.

Número de usuarios	Tiempo de respuesta promedio (s)
5	0.545
10	0.864
15	1.214
20	1.505
25	1.749
30	2.282

En la tabla 4.5 se muestra un resumen de los valores promedio en el consumo de recursos en los servidores de bases de datos.

Tabla 4.5. Consumo de recursos promedio en los servidores de bases de datos.

Recursos	Número de usuarios					
	5	10	15	20	25	30
% de uso de CPU	48.71	91.95	99.54	99.59	99.95	99.97
% de consumo de memoria RAM	5.99	6.26	7.70	9.06	10.33	11.58
% de uso de la red	3.84	5.10	5.65	5.66	5.67	6.01

En la tabla 4.6 se recogen los valores promedio del consumo de recursos en el balanceador de carga.

Tabla 4.6. Consumo de recursos promedio en el balanceador de carga.

Recursos	Número de usuarios					
	5	10	15	20	25	30
% de uso de CPU	0.62	0.74	0.78	0.75	0.75	0.76
% de consumo de memoria RAM	20.89	21.39	21.62	21.89	22.12	22.37
% de uso de la red	0.64	0.95	1.03	1.02	1.04	1.05

Análisis de los resultados

Como se puede apreciar en el gráfico de la figura 4.10 con el aumento de los niveles de concurrencia en el cluster de balanceo de carga para la capa de datos de la aplicación Registro de Unidades de Salud los tiempos de respuesta aumentan.

Esta prueba se realizó con el objetivo de tener una idea o poder estimar para determinada configuración de cluster cuales serían sus límites, ya sea por tiempos de respuesta extremadamente altos o por limitaciones en los recursos de los servidores.

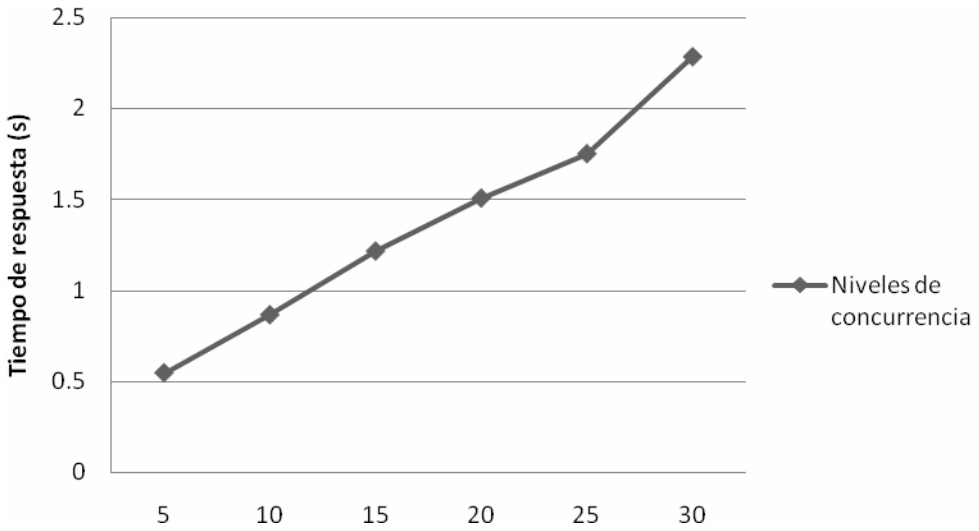

Figura 4.10. Tiempos de respuesta promedio contra niveles de concurrencia.

De manera equivalente a lo que pasa con los tiempos de respuesta, al aumentar los niveles de concurrencia, el consumo de recursos en los servidores bases de datos también aumenta, esto se puede observar en los gráficos de las figuras 4.11 y 4.12.

Como se puede apreciar además en la figura a partir de los quince usuarios concurrentes el consumo de CPU se mantiene por encima del 99%. Esto indica que con la configuración de cluster utilizada para la realización de las pruebas y los recursos de las computadoras utilizadas para el despliegue de dicha configuración, con 15 usuarios concurrentes ya el sistema está saturado.

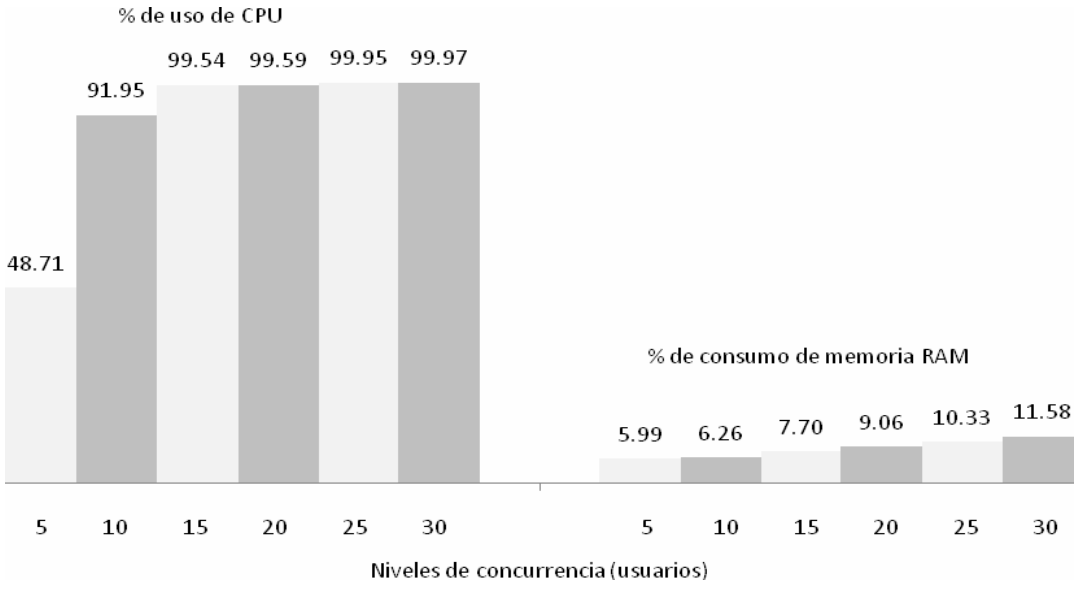

Figura 4.11. Consumo de recursos promedio (CPU, RAM) en los servidores de bases de datos.

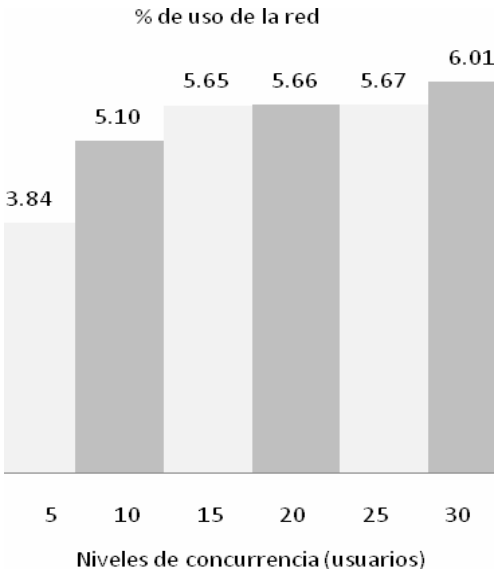

Figura 4.12. Consumo de recursos promedio (tráfico de red) en los servidores de bases de datos.

Con el aumento de los niveles de concurrencia el consumo de recursos en el balanceador también varía. Para este caso el consumo de CPU se mantiene casi constante pero el por ciento de uso de la memoria RAM y el por ciento de uso de la red van aumentando como se puede apreciar en las figuras 4.13 y 4.14.

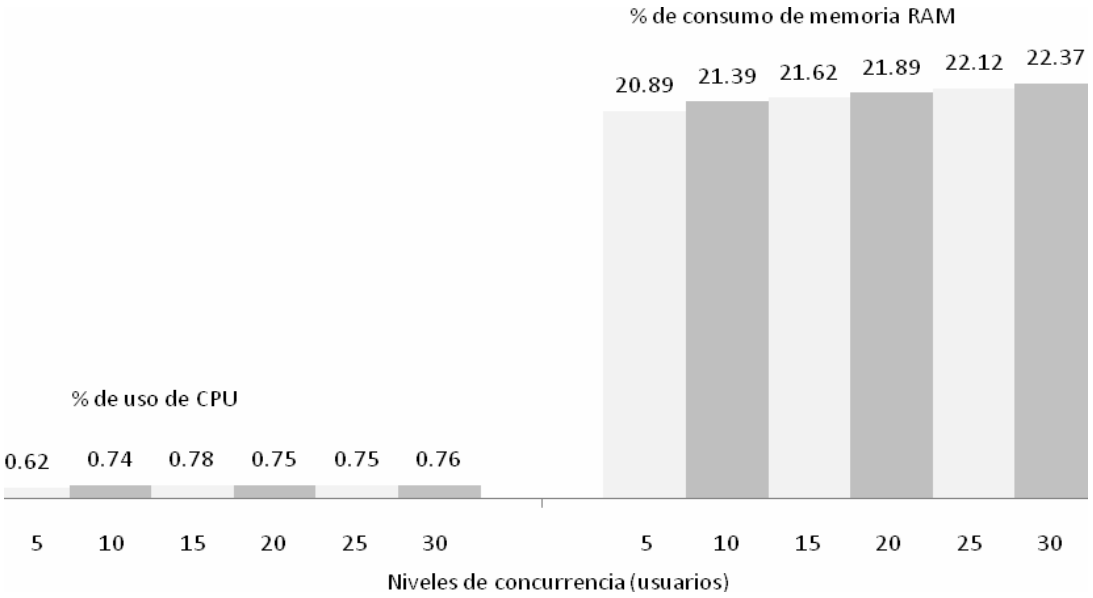

Figura 4.13. Consumo de recursos promedio (CPU, RAM) en el balanceador de carga.

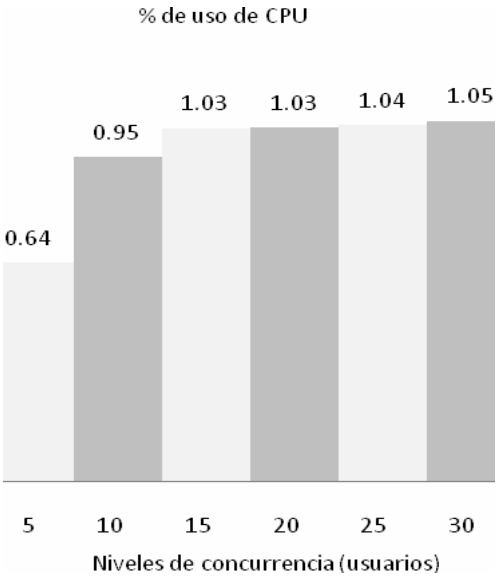

Figura 4.14. Consumo de recursos promedio (tráfico de red) en el balanceador de carga.

De manera semejante, con el aumento de los niveles de concurrencia, mientras el uso de la red se mantiene constante, el uso de CPU y el consumo de memoria RAM en el *master* de la replicación aumentan, pero para la prueba realizada no alcanzan niveles preocupantes, como se muestra en las figuras 4.15 y 4.16.

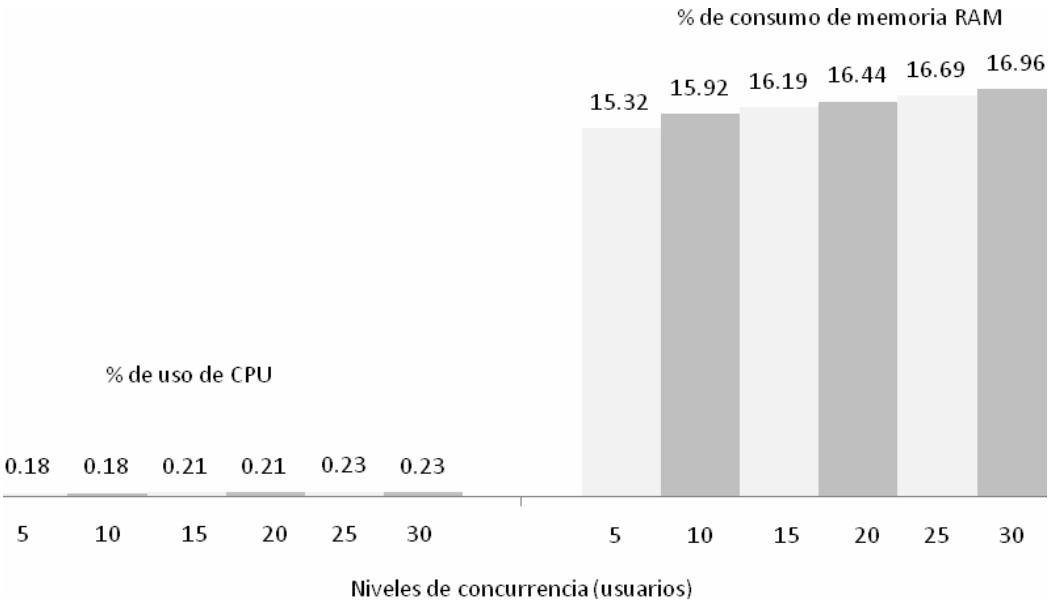

Figura 4.15. Consumo de recursos promedio (CPU, RAM) en el *master* de la replicación.

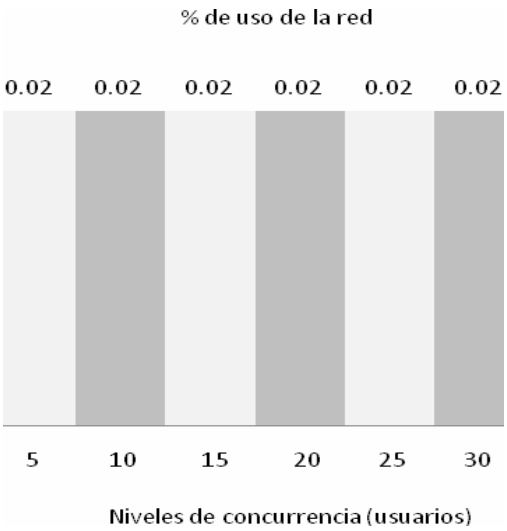

Figura 4.16. Consumo de recursos promedio (tráfico de red) en el *master* de la replicación.

4.3. Pruebas a la propuesta de solución Myosotis y Sequoia para RUS

4.3.1. Diseño de pruebas de configuración

Propósito de las pruebas

Comprobar que la propuesta de solución de cluster de bases de datos diseñada con Myosotis y Sequoia para la aplicación Registro de Unidades de Salud realiza todas las funcionalidades a probar correctamente.

Descripción del despliegue

Cluster de balanceo de carga para bases de datos que está compuesto por un nodo donde se encuentran las herramientas Myosotis y Sequoia y dos servidores de bases de datos. El cluster incluye funcionalidades de balanceo de carga para las consultas de lectura (select) y replicación para las consultas de modificación (insert, update, delete), así como, de recuperación ante fallos en los servidores de bases de datos.

Funcionalidades del sistema a probar

1. Balanceo de carga.
2. Replicación.
3. Recuperación ante fallos en los servidores de bases de datos.
4. Correcto funcionamiento de la aplicación.

Comportamiento esperado de las funcionalidades del sistema a probar

1. Balanceo de carga: Se espera que al llegar las consultas de lectura (select) al Sequoia este las distribuya entre los servidores de bases de datos de la forma más equitativa posible.
2. Replicación: Se espera que cuando se realice alguna operación de modificación sobre el Sequoia, la misma se realice además, sobre todos los servidores de bases de datos mediante el proceso de replicación.
3. Recuperación ante fallos de los servidores de bases de datos: Se espera que si uno de los servidores de bases de datos deja de prestar sus servicios el Sequoia lo detecte y deje de reenviarle consultas para que esto no afecte el funcionamiento del sistema.
4. Funcionamiento correcto de la aplicación: Se espera que el comportamiento de la aplicación al ejecutarse sobre la propuesta de solución diseñada para ella y sobre el despliegue original sea el mismo. Es decir, que la existencia del cluster sea transparente al usuario.

Planificación de la ejecución de las pruebas

1. Para analizar como el Sequoia distribuye las conexiones entre los servidores de bases de datos se va a realizar una navegación por la aplicación desde las computadoras clientes y en los servidores de bases de datos se va a analizar la cantidad de consultas que se realizaron sobre cada uno de ellos, esta información se va a extraer de los *logs* de acceso de los servidores MySQL.
2. Para analizar como se comporta el proceso de replicación se va a realizar una navegación por la aplicación desde las computadoras clientes, específicamente realizando operaciones de modificación, y posteriormente se va a analizar si las consultas que se realizaron sobre el Sequoia se realizaron a su vez sobre todos los servidores de bases de datos, esta información se va a extraer de los *logs* de acceso de los servidores MySQL.
3. Para probar la recuperación ante fallos de los servidores de bases de datos se va realizar una navegación en la aplicación desde las computadoras clientes y se le va a retirar el servicio de red uno de los servidores de bases de datos para comprobar que el sistema continúe brindando sus servicios.
4. La aplicación Registro de Unidades de Salud se desplegará en dos niveles, uno para la capa de presentación y lógica de negocio y otro para la capa de datos. De manera simultánea se realiza otro despliegue sobre el cluster de bases de datos. La prueba consiste en navegar en las aplicaciones de ambos despliegues y comparar el comportamiento de cada una prestando especial atención a los casos de uso representativos de la misma.

4.3.2. Recogida de datos y análisis de resultados de las pruebas de configuración

Durante el experimento se realizaron las navegaciones desde las computadoras clientes para comprobar todas las funcionalidades del sistema (balanceo de carga, replicación y recuperación ante fallos en los servidores de bases de datos) y que la aplicación funcionaba correctamente desplegada sobre el cluster obteniendo resultados positivos.

4.3.3. Diseño de pruebas de carga

Propósitos de las pruebas

1. Encontrar el número de usuarios concurrentes, que al navegar por la aplicación, provocan que el servidor de bases de datos, sobre el que esta se ejecuta, llegue al límite en el consumo de alguno de sus recursos.
2. Demostrar como al aumentar el número de servidores de bases de datos en el cluster de balanceo de carga, manteniendo el nivel de concurrencia constante, disminuye el tiempo de respuesta del sistema y el consumo de recursos en los servidores de bases de datos.
3. Demostrar como al aumentar el nivel de concurrencia en el cluster de balanceo de carga, manteniendo la cantidad de servidores de bases de datos constante, aumenta el tiempo de respuesta del sistema y el consumo de recursos en los servidores de bases de datos.

Descripción de los despliegues

1. La aplicación estará desplegada sobre dos servidores uno para la aplicación (capa de presentación y lógica de negocios) y otro para la base de datos (capa de datos).
2. La aplicación tendrá su capa de datos desplegada sobre un cluster de balanceo de carga. Este estará compuesto por un nodo donde se encuentra el Sequoia y el número de servidores de aplicación necesarios para la realización de la prueba. La capa de presentación y la lógica de negocio estarán sobre servidores externos al cluster.

Planificación de la ejecución de las pruebas

Desde tres computadoras diferentes se va a utilizar la herramienta JMeter para generar un nivel de concurrencia de usuarios navegando por la aplicación.

Después de un período de calentamiento de varios minutos para que la aplicación estabilice su funcionamiento, se realizará la medición para obtener el tiempo promedio que tarda el sistema en devolver las consultas solicitadas.

Para realizar la medición también se va a utilizar la herramienta JMeter pero desde una cuarta computadora. La herramienta se va a configurar para que realice la navegación de un usuario un total de 10 veces y además se le va a añadir un elemento al JMeter para la recogida del tiempo de respuesta del sistema para cada consulta que se le solicite. Este elemento se agrega porque, para los resultados de la prueba, es importante analizar los tiempos de respuesta promedio de un conjunto de páginas y no los tiempos de respuesta individuales de cada consulta.

Simultánea a la medición del tiempo de respuesta se realizará, utilizando la herramienta Sysstat, la medición del consumo de recursos en los servidores de bases de datos y en la computadora donde se encuentra Myosotis y Sequoia. La herramienta se ejecuta, a través de un script, con los parámetros necesarios para que cada dos segundos realice una medición hasta completar un total de 200.

Con el objetivo de encontrar el número de usuarios concurrentes que soporta un servidor de bases de datos en el despliegue original de la aplicación se comenzará la prueba con cinco usuarios y se va a repetir la prueba hasta encontrar el límite del servidor, añadiendo cinco usuarios concurrentes más en cada iteración de la misma.

Además, para poder demostrar como disminuyen los tiempos de respuesta promedio del sistema y el consumo de recursos en los servidores de bases de datos, la prueba será realizada variando la cantidad de nodos del cluster de balanceo de carga desde dos hasta seis servidores de bases de datos, manteniendo el mismo nivel de concurrencia.

Para poder demostrar como aumentan los tiempos de respuesta promedio del sistema y el consumo de recursos en los servidores de bases de datos, la prueba será realizada manteniendo la misma configuración de cluster (cluster de balanceo de carga con 3 servidores de bases de datos) con seis niveles de concurrencia diferentes.

4.3.4. Recogida de datos y análisis de resultados de las pruebas de carga

Datos recogidos

Para poder demostrar la disminución de los tiempos de respuesta del sistema y el del consumo de recursos en los servidores de bases de datos con el aumento de los nodos del cluster se recogieron los siguientes datos: tiempos de respuesta promedio, consumo de recursos promedio en los servidores de bases de datos y el consumo de recursos promedio en el balanceador de carga.

A continuación en la tabla 4.7 se muestran los tiempos de respuesta promedio del sistema.

Tabla 4.7. Tiempos de respuesta promedio del sistema.

Número de servidores de bases de datos fuera y dentro del cluster (c)	Tiempo de respuesta promedio (s)
1	1.592
2 (c)	1.312
3 (c)	1.220
4 (c)	1.116
5 (c)	1.064
6 (c)	1.007

En la tabla 4.8 se muestra un resumen de los valores promedio en el consumo de recursos en los servidores de bases de datos.

Tabla 4.8. Consumo de recursos promedio en los servidores de bases de datos.

Recursos	Número de servidores de bases de datos fuera y dentro del cluster (c)					
	1	2 (c)	3 (c)	4 (c)	5 (c)	6 (c)
% de uso de CPU	98.94	73.84	47.67	33.63	28.06	22.49
% de consumo de memoria RAM	12.53	11.67	11.66	11.34	10.91	10.86
% de uso de la red	0.43	0.34	0.24	0.26	0.31	0.33

En la tabla 4.9 se recogen los valores promedios del consumo de recursos en el balanceador de carga.

Tabla 4.9. Consumo de recursos promedio en el balanceador de carga.

Recursos	Número de servidores de bases de datos dentro del cluster (c)				
	2 (c)	3 (c)	4 (c)	5 (c)	6 (c)
% de uso de CPU	8.82	9.08	11.36	15.69	18.95
% de consumo de memoria RAM	23.34	25.11	26.00	26.65	27.77
% de uso de la red	1.85	2.08	2.84	3.95	4.86

Análisis de los resultados

Como se puede apreciar en la figura 4.17 la función representa la disminución de los tiempos de respuesta a medida que aumenta el número de nodos en el cluster. En el gráfico el punto aislado representa el tiempo de respuesta promedio de las consultas para un servidor de bases de datos fuera del cluster, simulando el despliegue original de la aplicación.

En el caso específico de esta solución es válido aclarar que no se realizó la comparación con un solo servidor en el cluster debido a que el método de replicación empleado para esta solución (RAIDb1), tiene como requisito que el cluster debe tener al menos dos nodos.

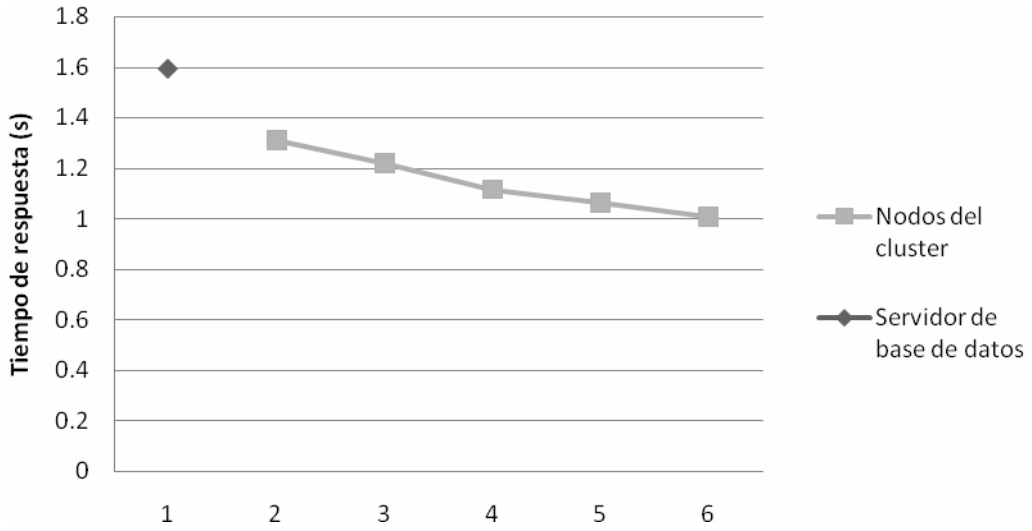

Figura 4.17. Tiempos de respuesta promedio contra número de nodos.

De manera semejante a lo que sucede con los tiempos de respuesta, al aumentar el número de nodos en el cluster, el consumo de recursos en los servidores de bases de datos disminuye. Esto se puede analizar en las figuras 4.18 y 4.19.

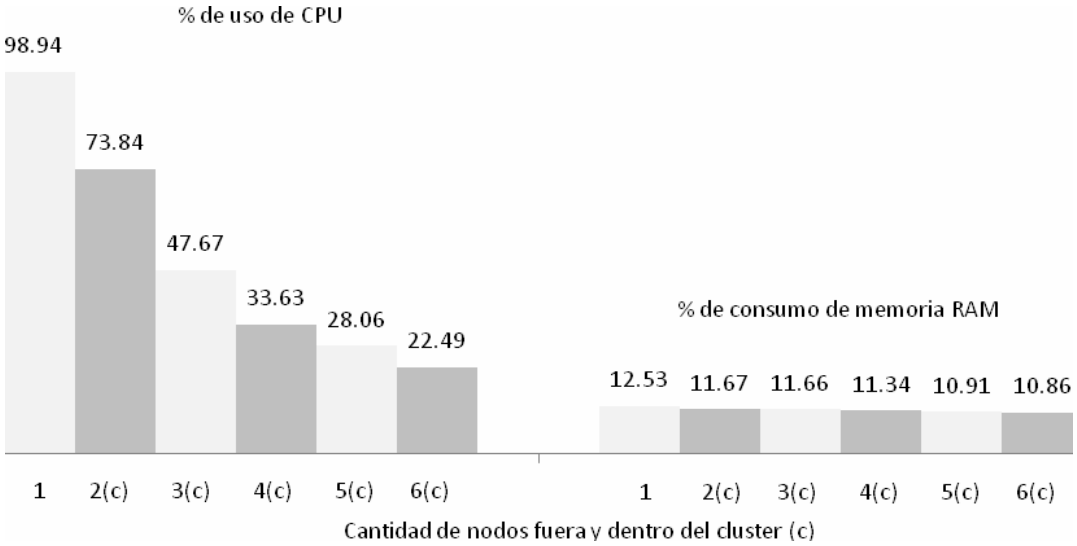

Figura 4.18. Consumo de recursos promedio (CPU, RAM) en los servidores de bases de datos.

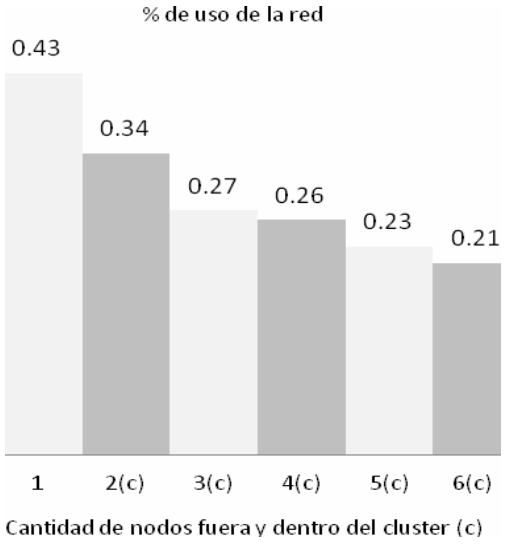

Figura 4.19. Consumo de recursos promedio (tráfico de red) en los servidores de bases de datos.

El aumento del número de nodos en el cluster trae como consecuencia un aumento en el consumo de recursos en el balanceador de carga. Aunque para la prueba realizada ninguno de estos alcanza valores preocupantes, como se muestra en las figuras 4.20 y 4.21.

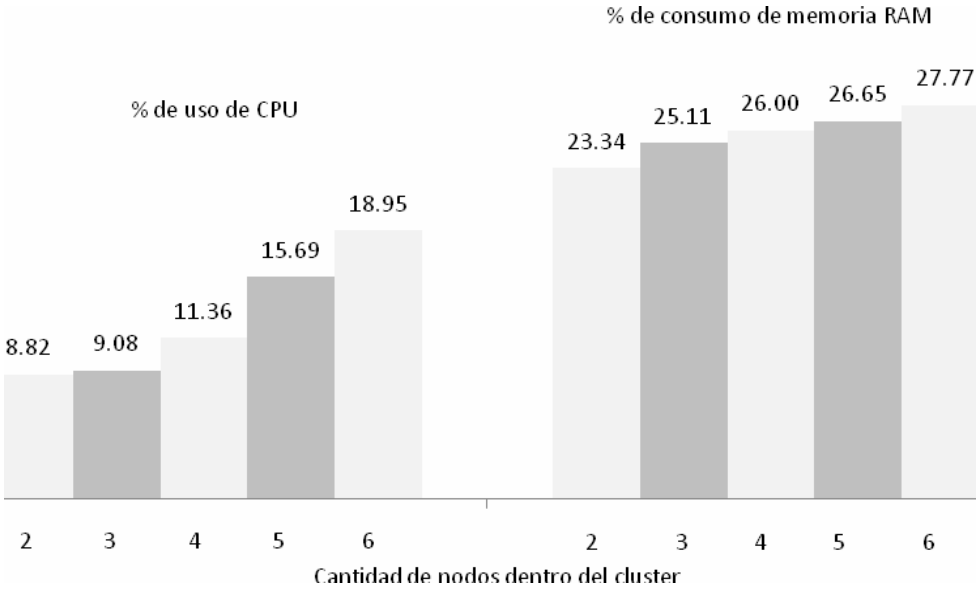

Figura 4.20. Consumo de recursos promedio (CPU, RAM) en el balanceador de carga.

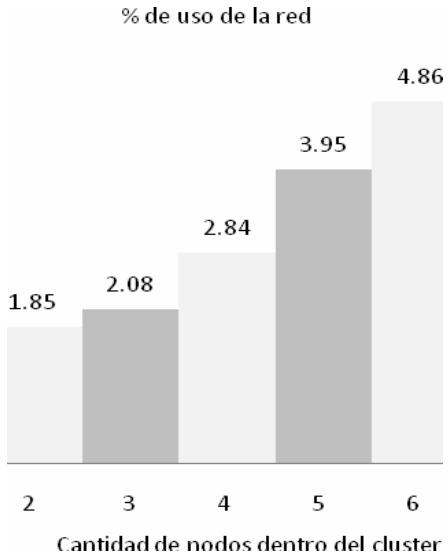

Figura 4.21. Consumo de recursos promedio (tráfico de red) en el balanceador de carga.

Lo más importante de analizar el consumo de recursos en el balanceador de carga es estimar el límite en el escalamiento del cluster, para así, tener una idea del número máximo de servidores de bases de datos que puede manejar el sistema.

Para hacer ese análisis se buscaron las líneas de tendencia para el consumo de cada de uno de los recursos que se están monitoreando y al graficarlas se obtiene el resultado que se muestra en la figura 4.22.

Además, es válido aclarar que, como se puede observar en la figura, el comportamiento de las curvas es muy accidentado. Debido a esto, la función que se obtuvo como resultado del ajuste de curvas que se realizó, utilizando el método lineal, no describe el comportamiento del consumo de los recursos e introduce un error de aproximación demasiado grande. Para lograr una función más acertada sería necesario continuar realizando el experimento para obtener más puntos en la gráfica, por lo que las funciones que se muestran en gráfico no son válidas y su único objetivo es mostrar la tendencia que tiene el consumo de recursos en el balanceador de carga.

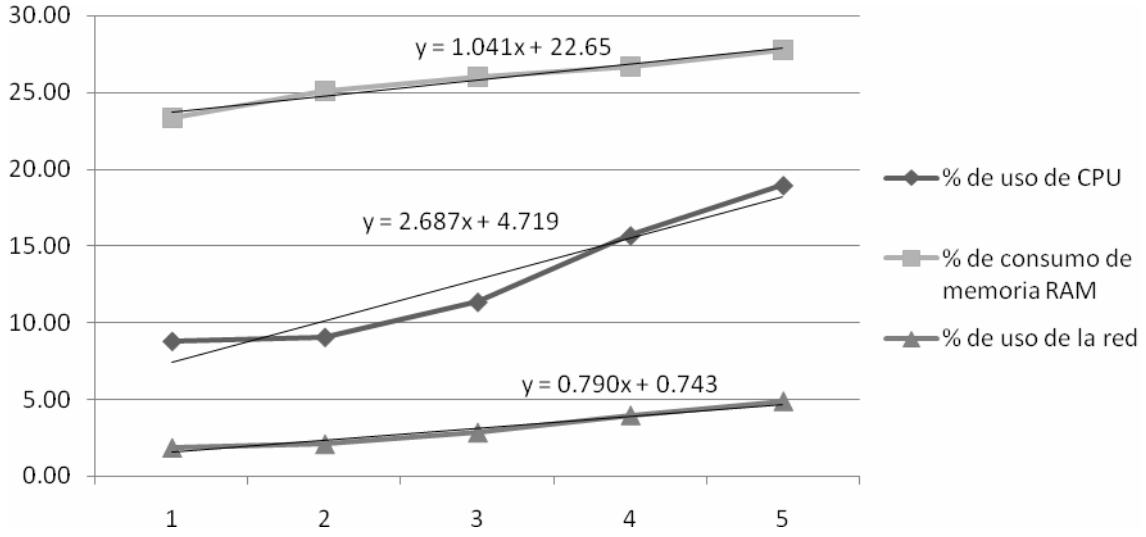

Figura 4.22. Líneas de tendencia del consumo de recursos para el balanceador de carga.

De manera análoga al análisis realizado para demostrar que con el aumento de los nodos del cluster disminuirían los tiempos de respuesta del sistema y el consumo de recursos en los servidores de bases de datos se realizaron pruebas para ver como se comportaba una de esas configuraciones a medida que iban aumentando los niveles de concurrencia.

Datos Recogidos

En la tabla 4.10 se muestran los tiempos de respuesta promedio del sistema.

Tabla 4.10. Tiempos de respuesta promedio del sistema.

Número de usuarios	Tiempo de respuesta promedio (s)
5	1.220
10	1.992
15	2.789
20	3.775
25	4.953
30	6.494

En la tabla 4.11 se muestra un resumen de los valores promedio en el consumo de recursos en los servidores de bases de datos.

Tabla 4.11. Consumo de recursos promedio en los servidores de bases de datos.

Recursos	Número de usuarios					
	5	10	15	20	25	30
% de uso de CPU	47.67	50.91	51.34	53.79	56.20	59.16
% de consumo de memoria RAM	11.76	12.34	12.52	12.94	13.57	15.52
% de uso de la red	0.24	0.28	0.42	0.62	0.74	0.95

En la tabla 4.12 se recogen los valores promedio del consumo de recursos en el balanceador de carga.

Tabla 4.12. Consumo de recursos promedio en el balanceador de carga.

Recursos	Número de usuarios					
	5	10	15	20	25	30
% de uso de CPU	9.08	10.26	14.22	18.42	22.86	26.37
% de consumo de memoria RAM	25.11	25.49	25.94	26.52	27.37	28.51
% de uso de la red	2.08	2.10	3.03	4.03	5.16	6.23

Análisis de los resultados

El análisis de los resultados que se brindan a continuación tiene como principal objetivo el de poder determinar cuáles serían los límites para la configuración del cluster, ya sea por tiempos de respuesta

extremadamente altos o limitaciones en los recursos de los servidores, el primero de estos parámetros en alcanzar su valor crítico.

En el gráfico 4.23 que se muestra a continuación la función representa la variación de los tiempos de respuesta promedio a medida que van aumentando los niveles de concurrencia.

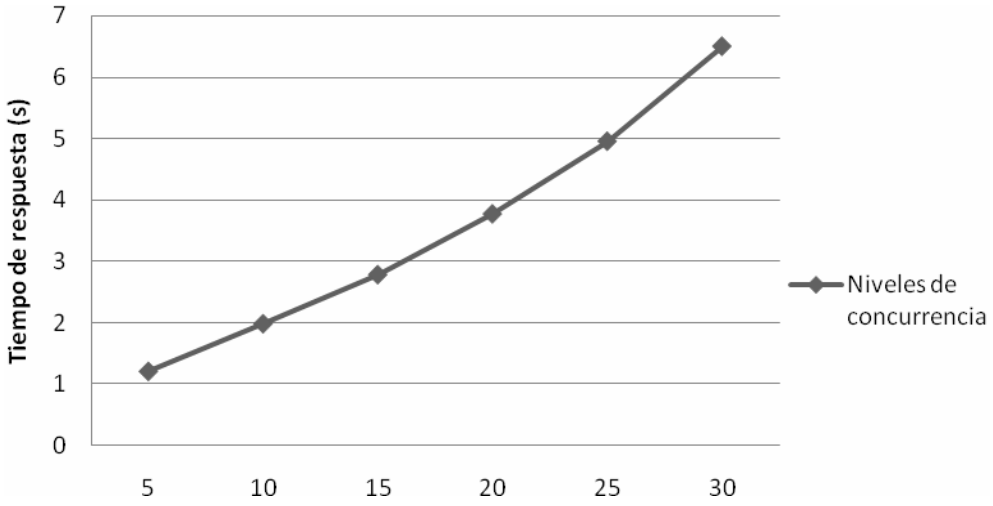

Figura 4.23. Tiempos de respuesta promedio contra niveles de concurrencia.

De manera equivalente a lo que pasa con los tiempos de respuesta, al aumentar los niveles de concurrencia, el consumo de recursos en los servidores bases de datos también aumenta, esto se puede observar en los gráficos de las figuras 4.24 y 4.25.

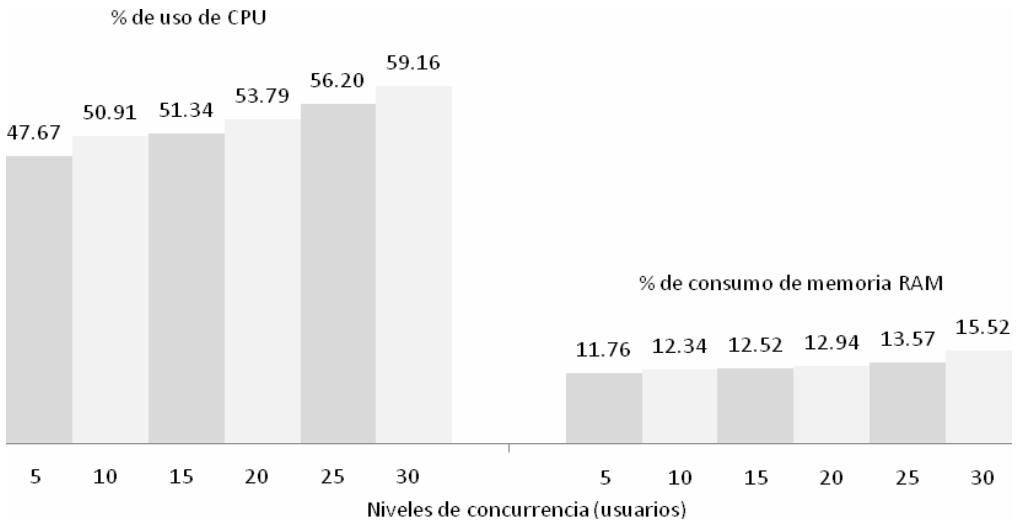

Figura 4.24. Consumo de recursos promedio (CPU, RAM) en los servidores de bases de datos.

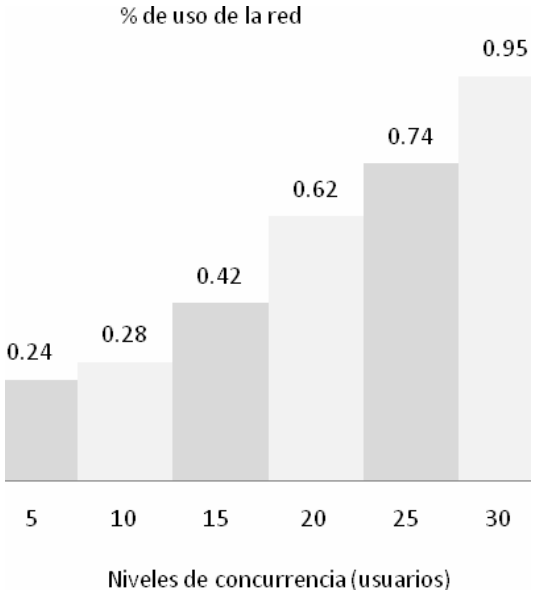

Figura 4.25. Consumo de recursos promedio (tráfico de red) en los servidores de bases de datos.

Con el aumento de los niveles de concurrencia el consumo de recursos en el balanceador también aumenta como se puede apreciar en las figuras 4.26 y 4.27.

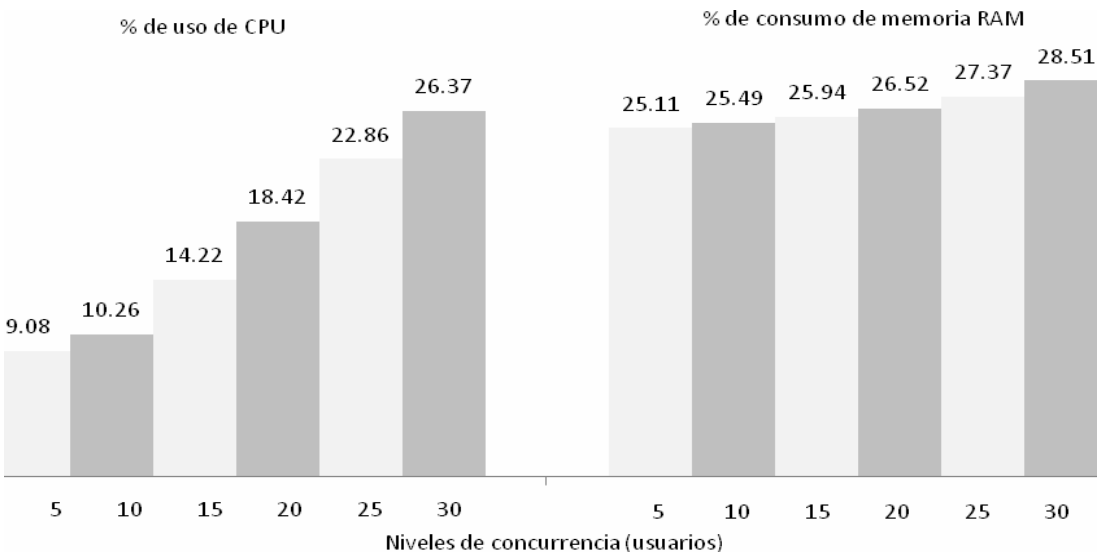

Figura 4.26. Consumo de recursos promedio (CPU, RAM) en el balanceador de carga.

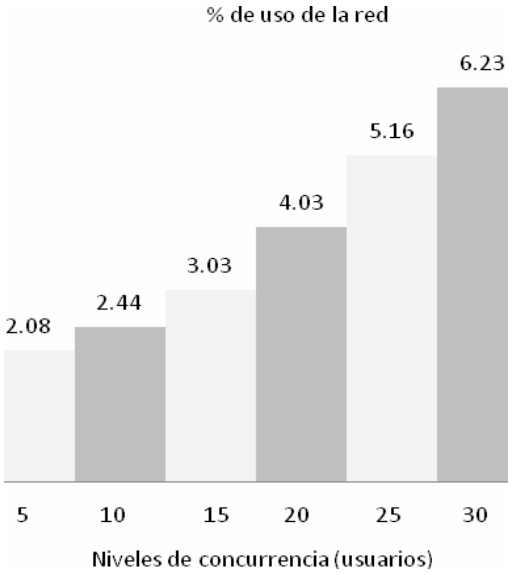

Figura 4.27. Consumo de recursos promedio (tráfico de red) en el balanceador de carga.

4.4. Pruebas a la propuesta de solución Sequoia para RCD

4.4.1. Diseño de pruebas de configuración

Propósito de las pruebas

Comprobar que la propuesta de solución de cluster de bases de datos diseñada con Sequoia para la aplicación Registro Centralizado de Donantes realiza todas las funcionalidades a probar correctamente.

Descripción del despliegue

Cluster de balanceo de carga para bases de datos que está compuesto por un nodo en el que se encuentra el Sequoia y dos servidores de bases de datos. El cluster incluye funcionalidades de balanceo de carga para las consultas de lectura (select) y replicación para las consultas de modificación (insert, update, delete), así como, de recuperación ante fallos en los servidores de bases de datos.

Funcionalidades del sistema a probar

1. Balanceo de carga.
2. Replicación.
3. Recuperación ante fallos en los servidores de bases de datos.

4. Correcto funcionamiento de la aplicación.

Comportamiento esperado de las funcionalidades del sistema a probar

1. Balanceo de carga: Se espera que al llegar las consultas al Sequoia este las distribuya entre los servidores de bases de datos de la forma más equitativa posible.
2. Replicación: Se espera que cuando se realice alguna operación de modificación la misma se realice sobre todos los servidores de bases de datos mediante el proceso de replicación.
3. Recuperación ante fallos de los servidores de bases de datos: Se espera que si uno de los servidores de bases de datos deja de prestar sus servicios el Sequoia lo detecte y deje de reenviarle consultas para que esto no afecte el funcionamiento del sistema.
4. Funcionamiento correcto de la aplicación: Se espera que el comportamiento de la aplicación al ejecutarse sobre la propuesta de solución diseñada para ella y sobre el despliegue original sea el mismo. Es decir, que la existencia del cluster sea transparente al usuario.

Planificación de la ejecución de las pruebas

1. Para analizar como el balanceador de carga distribuye las conexiones entre los servidores de bases de datos se va a realizar una navegación por la aplicación desde las computadoras clientes y en los servidores de bases de datos se va a analizar la cantidad de consultas que se realizaron sobre cada uno de ellos, esta información se va a extraer de los *logs* de acceso de los servidores MySQL.
2. Para analizar como se comporta el proceso de replicación se va a realizar una navegación por la aplicación desde las computadoras clientes, específicamente realizando operaciones de modificación, y posteriormente se va a analizar si las consultas que se realizaron sobre el Sequoia se realizaron a su vez sobre todos los servidores de bases de datos, esta información se va a extraer de los *logs* de acceso de los servidores MySQL.
3. Para probar la recuperación ante fallos de los servidores de bases de datos se va realizar una navegación en la aplicación desde las computadoras clientes y se le va a retirar el servicio de red uno de los servidores de bases de datos para comprobar que el sistema continúe brindando sus servicios.
4. La aplicación Registro de Unidades de Salud se desplegará en dos niveles, uno para la capa de presentación y lógica de negocio y otro para la capa de datos. De manera simultánea se realiza otro despliegue sobre el cluster de bases de datos. La prueba consiste en navegar en las aplicaciones

de ambos despliegues y comparar el comportamiento de cada una prestando especial atención a los casos de uso representativos de la misma.

4.4.2. Recogida de datos y análisis de resultados de las pruebas de configuración

Durante el experimento se realizaron las navegaciones desde las computadoras clientes para comprobar todas las funcionalidades del sistema (balanceo de carga, replicación y recuperación ante fallos en los servidores de bases de datos) y que la aplicación funcionaba correctamente desplegada sobre el cluster obteniendo resultados positivos.

4.4.3. Diseño de pruebas de carga

Propósitos de las pruebas

1. Encontrar el número de usuarios concurrentes, que al navegar por la aplicación, provocan que el servidor de bases de datos, sobre el que esta se ejecuta, llegue al límite en el consumo de alguno de sus recursos.
2. Demostrar como al aumentar el número de servidores de bases de datos en el cluster de balanceo de carga, manteniendo el nivel de concurrencia constante, disminuye el tiempo de respuesta del sistema y el consumo de recursos en los servidores de bases de datos.
3. Demostrar como al aumentar el nivel de concurrencia en el cluster de balanceo de carga, manteniendo la cantidad de servidores de bases de datos constante, aumenta el tiempo de respuesta del sistema y el consumo de recursos en los servidores de bases de datos.

Descripción de los despliegues

1. La aplicación estará desplegada sobre dos servidores uno para la aplicación (capa de presentación y lógica de negocios) y otro para la base de datos (capa de datos).
2. La aplicación tendrá su capa de datos desplegada sobre un cluster de balanceo de carga. Este estará compuesto por un nodo donde se encuentra el Sequoia, y el número de servidores de aplicación necesarios para la realización de la prueba. La capa de presentación y la lógica de negocio estarán sobre servidores externos al cluster.

Planificación de la ejecución de las pruebas

Desde tres computadoras diferentes se va a utilizar la herramienta JMeter para generar un nivel de concurrencia de usuarios navegando por la aplicación.

Después de un período de calentamiento de varios minutos para que la aplicación estabilice su funcionamiento, se realizará la medición para obtener el tiempo promedio que tarda el sistema en devolver las consultas solicitadas.

Para realizar la medición también se va a utilizar la herramienta JMeter pero desde una cuarta computadora. La herramienta se va a configurar para que realice la navegación de un usuario un total de 10 veces y además se le va a añadir un elemento al JMeter para la recogida del tiempo de respuesta del sistema para cada consulta que se le solicite. Este elemento se agrega porque, para los resultados de la prueba, es importante analizar los tiempos de respuesta promedio de un conjunto de páginas y no los tiempos de respuesta individuales de cada consulta.

Simultánea a la medición del tiempo de respuesta se realizará, utilizando la herramienta Sysstat, la medición del consumo de recursos en los servidores de bases de datos y en la computadora del Sequoia. La herramienta se ejecuta, a través de un script, con los parámetros necesarios para que cada dos segundos realice una medición hasta completar un total de 200.

Con el objetivo de encontrar el número de usuarios concurrentes que soporta un servidor de bases de datos en el despliegue original de la aplicación se comenzará la prueba con cinco usuarios y se va a repetir la prueba hasta encontrar el límite del servidor, añadiendo cinco usuarios concurrentes más en cada iteración de la misma.

Además, para poder demostrar como disminuyen los tiempos de respuesta promedio del sistema y el consumo de recursos en los servidores de bases de datos, la prueba será realizada variando la cantidad de nodos del cluster de balanceo de carga desde dos hasta seis servidores de bases de datos, manteniendo el mismo nivel de concurrencia.

Para poder demostrar como aumentan los tiempos de respuesta promedio del sistema y el consumo de recursos en los servidores de bases de datos, la prueba será realizada manteniendo la misma configuración de cluster (cluster de balanceo de carga con 3 servidores de bases de datos) con seis niveles de concurrencia diferentes.

4.4.4. Recogida de datos y análisis de resultados de las pruebas de carga

Datos recogidos

Para poder demostrar la disminución de los tiempos de respuesta del sistema y el del consumo de recursos en los servidores de bases de datos con el aumento de los nodos del cluster se recogieron los siguientes datos: tiempos de respuesta promedio, consumo de recursos promedio en los servidores de bases de datos y el consumo de recursos promedio en el balanceador de carga.

A continuación en la tabla 4.13 se muestran los tiempos de respuesta promedio del sistema.

Tabla 4.13. Tiempos de respuesta promedio del sistema.

Número de servidores de bases de datos fuera y dentro del cluster (c)	Tiempo de respuesta promedio (s)
1	2.420
2 (c)	3.386
3 (c)	3.400
4 (c)	3.448
5 (c)	3.470
6 (c)	3.490

En la tabla 4.14 se muestra un resumen de los valores promedio en el consumo de recursos en los servidores de bases de datos.

Tabla 4.14. Consumo de recursos promedio en los servidores de bases de datos.

Recursos	Número de servidores de bases de datos fuera y dentro del cluster (c)					
	1	2 (c)	3 (c)	4 (c)	5 (c)	6 (c)
% de uso de CPU	89.36	24.89	15.16	11.52	9.25	7.66
% de consumo de memoria RAM	49.23	37.59	37.82	37.91	38.08	8.65
% de uso de la red	29.54	9.67	6.29	4.69	3.73	3.09

En la tabla 4.15 se recogen los valores promedios del consumo de recursos en el balanceador de carga.

Tabla 4.15. Consumo de recursos promedio en el balanceador de carga.

Recursos	Número de servidores de bases de datos dentro del cluster (c)				
	2 (c)	3 (c)	4 (c)	5 (c)	6 (c)
% de uso de CPU	92.06	90.55	91.07	90.95	91.18
% de consumo de memoria RAM	20.36	19.95	19.41	19.51	23.31
% de uso de la red	54.12	53.53	53.55	53.44	53.36

Análisis de los resultados

Durante los experimentos realizados se pudo probar que la aplicación RCD (Registro Centralizado de Donantes) tiene la característica de que cada caso de uso fue implementado utilizando un número significativamente grande de consultas muy sencillas, en su mayoría del tipo *select.* Algunas de las funcionalidades más representativas realizan más de nueve mil consultas a las bases de datos, como por ejemplo los casos de uso que implican búsqueda de datos.

Estas consultas tan sencillas son muy fáciles de procesar por los servidores de bases de datos y casi no constituyen una carga de trabajo para estos.

Cuando se implementa un cluster de balanceo de carga un factor importante a tener en cuenta es el rendimiento del balanceador de carga. Normalmente un balanceador de carga debe realizar un consumo modesto de recursos y dejar toda la carga de procesamiento para los servidores reales sobre los cuales se realiza el balanceo. Esto quiere decir que el cluster será capaz de procesar información hasta que los recursos de los servidores reales se vean agotados pudiendo estar sobrecargados los servidores sin que el servidor lo esté.

Cuando se alcanza el límite de estos recursos puede añadirse un nuevo nodo al cluster, de esta forma el sistema permanece operacional y continúa prestando sus servicios de manera eficiente.

Al recibir una petición realizada por un cliente, Sequoia determina hacia que servidor de bases de datos de la envía. Para lograr esto se realiza un *parsing* (análisis sintáctico) de las consultas recibidas lo cual le permite identificar si dicha consulta es de lectura, en cuyo caso la redirecciona hacia un servidor en específico, o si es una consulta de escritura la cual envía hacia todos los servidores o hacia un grupo de ellos, en dependencia del método de replicación que se utilice.

El *parsing* (análisis sintáctico) de las consultas requiere de un procesamiento adicional en el balanceador y de la utilización de algunos de sus recursos como: memoria RAM (*Random Access Memory* / Memoria de Acceso Aleatorio), la cual es utilizada para mantener en *cache* las consultas que ya fueron analizadas, y tiempo de CPU (*Central Proccessing Unit* / Unidad Central de Procesamiento).

Cuando se realizan una cantidad pequeña de consultas complejas el *parsing* de las mismas requiere de un menor consumo de recursos en el balanceador y el mayor volumen de procesamiento se realiza en los servidores de bases de datos.

Para aplicaciones como RCD (Registro Centralizado de Donantes) que realizan, atípicamente, una gran cantidad de consultas sencillas a las bases de datos, el proceso de *parsing* en el balanceador bajo condiciones de alta concurrencia se vuelve bastante complejo. La complejidad viene dada debido

a que el *parsing* debe realizarse sobre una gran cantidad de consultas que no representan carga apenas para los servidores de bases de datos, los cuales envían sus respuestas en un tiempo proporcionalmente menor al requerido por el que invierte el balanceador para realizar esta acción.

El incremento del número de nodos del cluster unido al *parsing* de las consultas provoca un aumento del consumo de recursos en el servidor sobre el que este se ejecuta, lo que trae como consecuencia que se deterioren los tiempos de respuesta del sistema. Las figuras 4.28 y 4.29 representan el consumo de recursos del balanceador y las figuras 4.30 y 4.31 representan el consumo de recursos en los servidores de bases de datos durante el experimento.

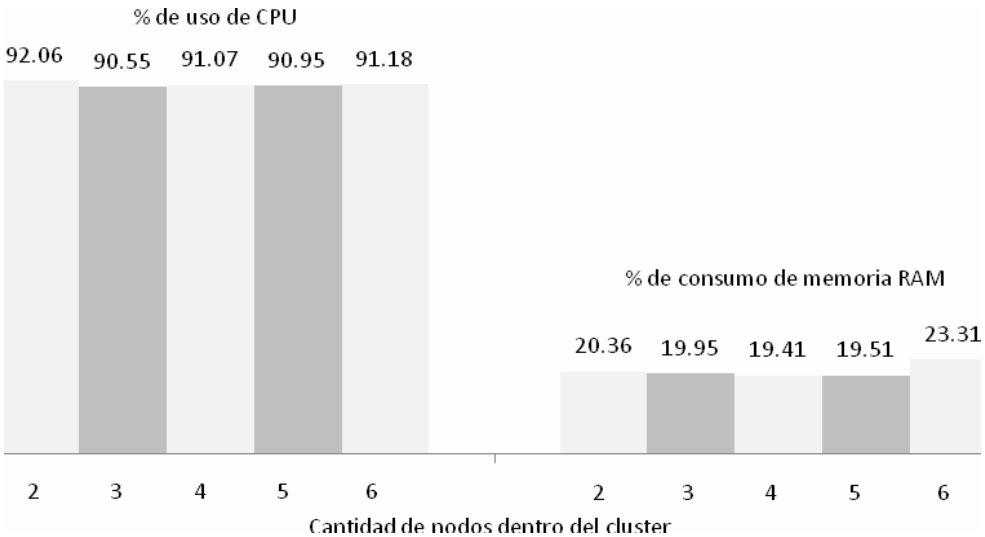

Figura 4.28. Consumo de recursos promedio (CPU, RAM) en el balanceador de carga.

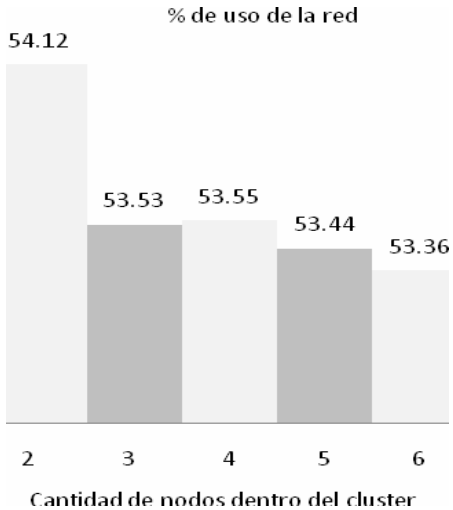

Figura 4.29. Consumo de recursos promedio (tráfico de red) en el balanceador de carga.

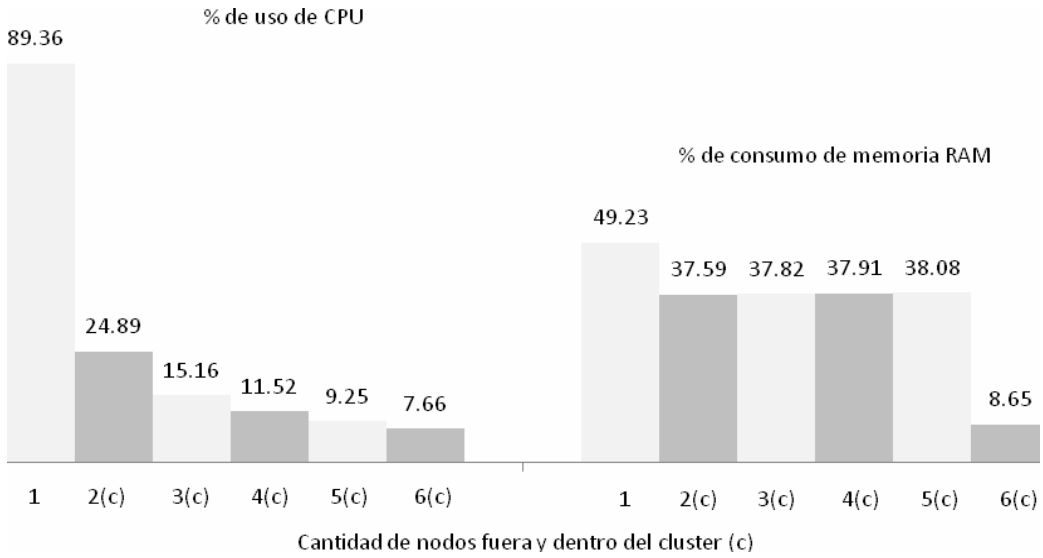

Figura 4.30. Consumo de recursos promedio (CPU, RAM) en los servidores de bases de datos.

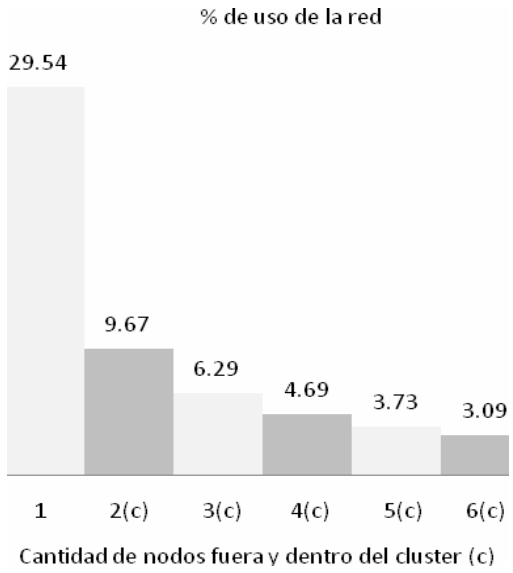

Figura 4.31. Consumo de recursos promedio (tráfico de red) en los servidores de bases de datos.

Los tiempos de respuesta obtenidos en todas las configuraciones del cluster fueron más elevados que el tiempo en que fue capaz de responder un solo servidor, figura 4.32. Se puedo concluir que las demoras de los tiempos fueron introducidas por el balanceador de carga el cual utilizó un alto porciento de consumo de CPU, figura 4.28, durante todo el tiempo de realización de la prueba.

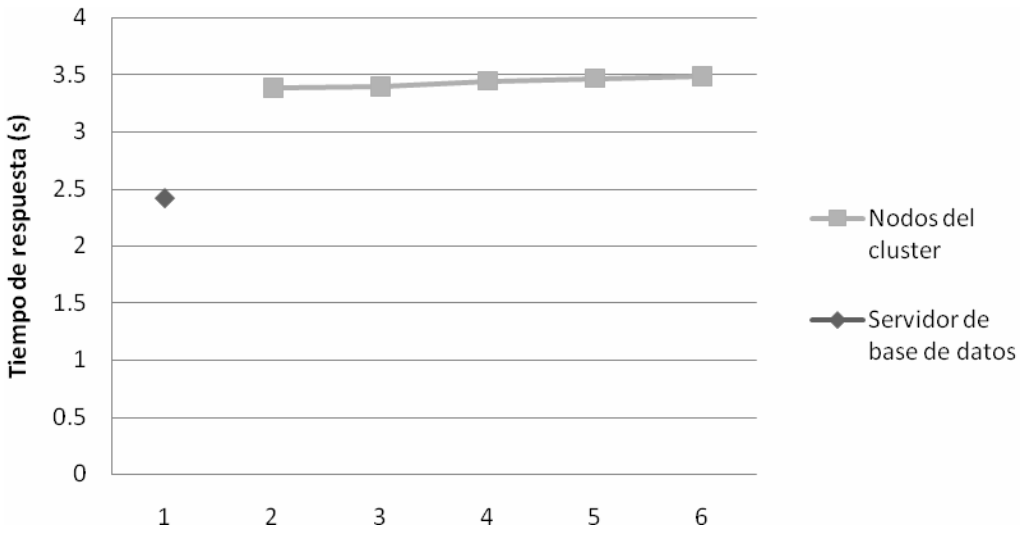

Figura 4.32. Tiempos de respuesta promedio.

Conclusiones de la prueba: Los resultados alcanzados durante las pruebas de carga mostraron que el cluster de bases de datos Sequoia no constituye una solución válida para las aplicaciones con características semejantes a RCD (Registro Centralizado de Donantes).

4.5. Pruebas a la propuesta de solución Myosotis y Sequoia para BM

4.5.1. Diseño de pruebas de configuración

Propósito de las pruebas

Comprobar que la propuesta de solución de cluster de bases de datos diseñada con Myosotis y Sequoia para la aplicación Balance Material realiza todas las funcionalidades a probar correctamente.

Descripción del despliegue

Cluster de balanceo de carga para bases de datos que está compuesto por un nodo donde se encuentra Myosotis y Sequoia y dos servidores de bases de datos. El cluster incluye funcionalidades de balanceo de carga para las consultas de lectura (select) y replicación para las consultas de modificación (insert, update, delete), así como, de recuperación ante fallos en los servidores de bases de datos.

Funcionalidades del sistema a probar

1. Balanceo de carga.
2. Replicación.

3. Recuperación ante fallos en los servidores de bases de datos.
4. Correcto funcionamiento de la aplicación.

Comportamiento esperado de las funcionalidades del sistema a probar

1. Balanceo de carga: Se espera que al llegar las consultas de lectura (select) al Sequoia este las distribuya entre los servidores de bases de datos de la forma más equitativa posible.
2. Replicación: Se espera que cuando se realice alguna operación de modificación la misma se realice sobre todos los servidores de bases de datos mediante el proceso de replicación.
3. Recuperación ante fallos de los servidores de bases de datos: Se espera que si uno de los servidores de bases de datos deja de prestar sus servicios el balanceador de carga lo detecte y deje de reenviarle consultas para que esto no afecte el funcionamiento del sistema.
4. Funcionamiento correcto de la aplicación: Se espera que el comportamiento de la aplicación al ejecutarse sobre la propuesta de solución diseñada para ella y sobre el despliegue original sea el mismo. Es decir, que la existencia del cluster sea transparente al usuario.

Planificación de la ejecución de las pruebas

1. Para analizar como el Sequoia distribuye las conexiones entre los servidores de bases de datos se va a realizar una navegación por la aplicación desde las computadoras clientes y en los servidores de bases de datos se va a analizar la cantidad de consultas que se realizaron sobre cada uno de ellos, esta información se va a extraer de los *logs* de acceso de los servidores MySQL.
2. Para analizar como se comporta el proceso de replicación se va a realizar una navegación por la aplicación desde las computadoras clientes, específicamente realizando operaciones de modificación, y posteriormente se va a analizar si las consultas que se realizaron sobre el Sequoia se realizaron a su vez sobre todos los servidores de bases de datos, esta información se va a extraer de los *logs* de acceso de los servidores MySQL.
3. Para probar la recuperación ante fallos de los servidores de bases de datos se va realizar una navegación en la aplicación desde las computadoras clientes y se le va a retirar el servicio de red uno de los servidores de bases de datos para comprobar que el sistema continúe brindando sus servicios.
4. La aplicación Registro de Unidades de Salud se desplegará en dos niveles, uno para la capa de presentación y lógica de negocio y otro para la capa de datos. De manera simultánea se realiza otro despliegue sobre el cluster de bases de datos. La prueba consiste en navegar en las aplicaciones

de ambos despliegues y comparar el comportamiento de cada una prestando especial atención a los casos de uso representativos de la misma.

4.5.2. Recogida de datos y análisis de resultados de las pruebas de configuración

Durante la primera iteración del experimento utilizando la versión 2.10.10 del *software* Sequoia se encontraron algunos errores cuando se realizó la comparación del funcionamiento de la aplicación Balance Material sobre el cluster y sobre su despliegue original. Los errores fueron causados cuando se realizaron llamadas a los procedimientos almacenados utilizados por la aplicación.

La versión de Sequoia 2.10.10 trata los procedimientos almacenados a nivel de *backend,* esto trae consigo dificultades en el *parsing* (análisis sintáctico) de las consultas y por consiguiente errores en el proceso de replicación. En las pruebas realizadas con un método de replicación RAIDb1 para bases de datos MySQL 5.0.1 se detectó que al realizar consultas de modificación (insert, update, delete) estas sólo se replican en uno de los servidores de bases de datos cuando deberían hacerlo en todos.

En una segunda iteración del experimento se utilizó la versión 3.0 beta 2. Esta versión trata los procedimientos almacenados a nivel de *virtual database* (base de datos virtual) lo cual resuelve los errores mencionados anteriormente. Pero, durante la prueba realizada, se detectaron errores con el algoritmo de balanceo de carga (RAIDb1) por lo que los resultados fueron negativos también.

Los resultados alcanzados durante las pruebas de configuración demostraron que el cluster de bases de datos diseñado con Myosotis y Sequoia no constituye una solución funcional para las aplicaciones con características semejantes a Balance Material que manejan procedimientos almacenados, sin embargo, esta solución no debe ser descartada completamente y se debe dar seguimiento a la liberación de nuevas versiones de Sequoia, en especial la versión 4.0 que, según sus desarrolladores, será diseñada a partir de la versión estable 2.10.10 e implementará el manejo de los procedimientos almacenado a nivel de base de datos virtual como se realiza en la versión 3.0.

4.6. Análisis de resultados de las pruebas de benchmark

El experimento se realizó con el objetivo de buscar entre la solución de cluster diseñada con LVS (*Linux Virtual Server*) y replicación y la solución diseñada con Myosotis y Sequoia, cual es la óptima para la aplicación Registro de Unidades de Salud.

En la figura 4.33 que se presenta a continuación se muestran las gráficas de variación de los tiempos de respuestas a medida que aumenta el número de nodos para ambas soluciones y se muestra además el tiempo de respuesta para un servidor de bases de datos fuera del cluster. Como se puede apreciar para la solución de LVS y Replicación los tiempos de respuesta son menores.

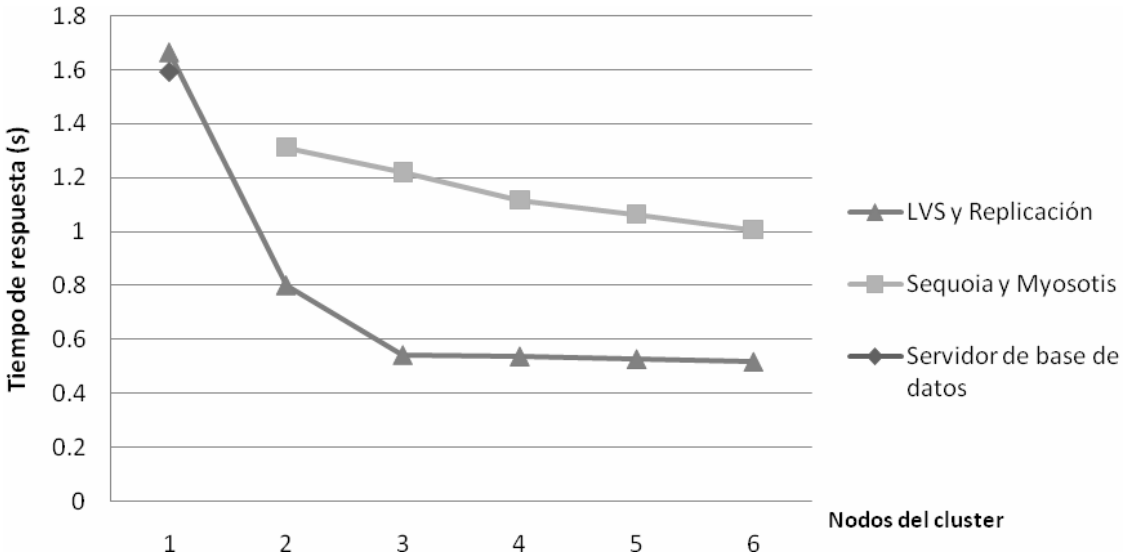

Figura 4.33. Tiempos de respuesta promedio.

Para determinar los tiempos de respuesta promedio para ambas soluciones se eligió una configuración de cluster con tres servidores reales. Los resultados obtenidos se muestran en la figura 4.34.

Como se puede observar para los mismos niveles de concurrencia en la solución diseñada con Myosotis y Sequoia los tiempos de respuesta promedio se van deteriorando más rápido y para 30 usuarios concurrentes ya alcanzan valores de 6.494 segundos contra solo 2.282 de la solución diseñada con LVS y Replicación.

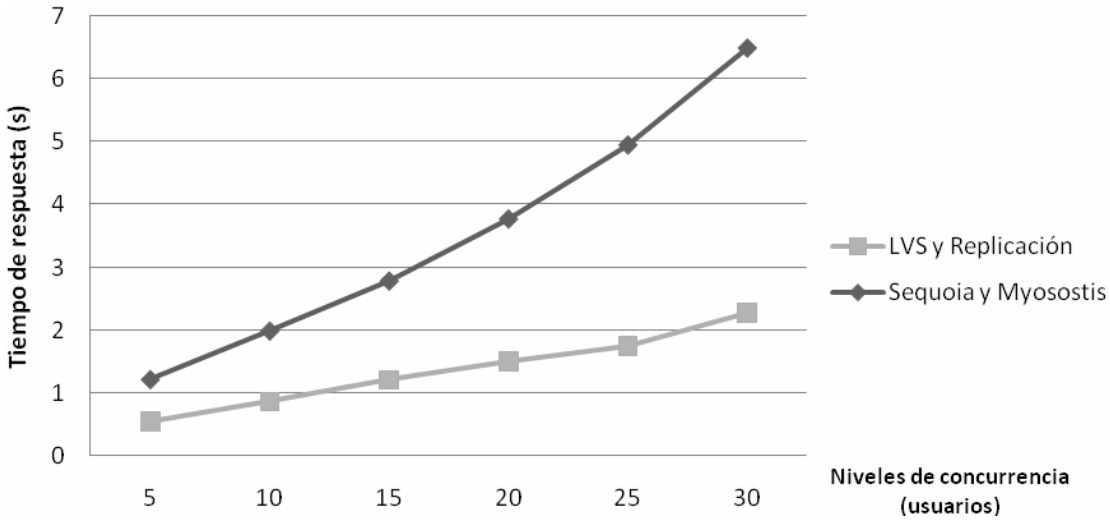

Figura 4.34. Tiempos de respuesta promedio contra niveles de concurrencia.

Los resultados obtenidos durante la prueba de *benchmark* demostraron que la propuesta de solución de cluster de bases de datos diseñada con LVS (*Linux Virtual Server*) y replicación MySQL es la óptima para las aplicaciones con características similares a Registro de Unidades de Salud.

4.7. Conclusiones

Las pruebas se realizaron para validar cada una de las propuestas de soluciones diseñadas anteriormente. Cada una de estas propuestas de solución fue sometida a dos tipos de pruebas: pruebas de configuración y pruebas de carga. Las pruebas de configuración se hicieron con el objetivo de comprobar que las soluciones de *cluster* no afectan el funcionamiento de las aplicaciones, además se comprobó el funcionamiento correcto del sistema. Las pruebas de carga se realizaron principalmente para medir los tiempos de respuesta una vez que las aplicaciones se encontraran desplegadas sobre las soluciones de *cluster.* Además se realizó un análisis del consumo de recursos de los nodos integrantes del cluster.

CONCLUSIONES

En el presente trabajo se realizó un estudio de las aplicaciones desarrolladas por la Universidad de las Ciencias Informáticas (UCI) y la empresa Softel para el sector de la salud en Cuba y se diseñaron varias soluciones de cluster de bases de datos para ellas, dando cumplimiento a los objetivos definidos en la investigación.

Durante las pruebas realizadas para validar las soluciones propuestas se detectó que algunas de ellas no ofrecieron los resultados esperados. En este caso se encontró la solución de cluster Myosotis y Sequoia para las aplicaciones con característica semejantes a Balance Material. La solución presentó problemas que afectaron el funcionamiento de la aplicación por lo que fue necesario descartarla durante las pruebas de configuración. Por los resultados obtenidos se concluyó que la versión del software Sequoia no está lista aun para esta solución.

Por otra parte la propuesta de solución Sequoia para aplicaciones del grupo representado por Registro Centralizado de Donantes presentó problemas durante las pruebas de rendimiento. El cluster propuesto no redujo los tiempos de respuesta de las consultas realizadas y esto se debió principalmente a algunas ineficiencias detectadas en la implementación de la lógica de negocio de las aplicaciones de este grupo. Por tal motivo también hubo que prescindir de esta solución.

De esta forma sólo se pudieron encontrar dos soluciones de cluster válidas para aplicaciones del primer grupo representado por Registro de Unidades de Salud. Estas soluciones fueron: el cluster confeccionado utilizando las técnicas de LVS más la replicación de MySQL y el cluster compuesto por Myosotis y Sequoia. Se realizó además una prueba de *benchmark* a ambas propuestas teniendo en cuenta los tiempos de respuesta obtenidos durante la variación de concurrencia de usuarios y la variación de nodos dentro del cluster. Los menores tiempos fueron alcanzados con la solución de cluster compuesta por LVS y la replicación de MySQL.

Para esta solución se propuso además un mecanismo de alta disponibilidad en el balanceador de carga y un procedimiento para realizar la recuperación del servidor maestro (*master*) de la replicación ante la caída del mismo.

RECOMENDACIONES

Una vez cumplido el objetivo general de la investigación, se recomienda:

1. Implementar totalmente, en todas las aplicaciones, el requisito no funcional de separación de consultas de lectura y escritura para una configuración de replicación *master*-esclavo. Este requisito está especificado en el documento de arquitectura confeccionado de manera conjunta por la Universidad de las Ciencias Informáticas y Softel titulado “Arquitectura, normas y tecnologías para el desarrollo de aplicaciones informáticas para la Salud Pública en Cuba.” (Softel 2007)
2. Eliminar las ineficiencias en el código de implementación de las aplicaciones. Estas ineficiencias vienen dadas por el uso excesivo de consultas en la implementación de algunas funcionalidades de la aplicación.
3. Proponer un mecanismo de alta disponibilidad para las soluciones de cluster que incluyen el controlador Sequoia aprovechando algunas de las facilidades brindadas por el mismo a este efecto.
4. Dar seguimiento a la liberación de futuras versiones de la herramienta Sequoia, en especial la versión 4.0 que, según sus desarrolladores, será diseñada a partir de la versión estable 2.10.10 e implementará el manejo de los procedimientos almacenado a nivel de base de datos virtual como se realiza en la versión 3.0.
5. Perfeccionar la concepción y elaborar un procedimiento para el despliegue de las soluciones de cluster.

REFERENCIAS BIBLIOGRÁFICAS

1. Andreasson, O. (2007). "Iptables tutorial 1.2.2." Retrieved 2008 Junio, from http://www.netfilter.org/.
2. Apache Jakarta Project. (2008). "JMeter User's Manual." Retrieved 2008 Mayo, from http://jakarta.apache.org/jmeter/usermanual/index.html.
3. Balling, D. J. and J. Zawodny (2004). Chapter 7. Replication. High Performance MySQL. Sebastopol, O'Reilly**:** 294.
4. Barroso, L. A., J. Dean, et al. (2003). "Web search for a planet: The Google cluster architecture." Retrieved 2008 Marzo, from http://labs.google.com/papers/googlecluster-ieee.pdf.
5. BLSF Equipo de desarrollo (2005). Chapter 11. Utilidades del Sistema. Sysstat-5.1.5. Más Allá de Linux.
6. Bourke, T. (2001). Server Load Balancing. Sebastopol, O'Reilly & Associates, Inc.
7. Cecchet, E., J. Marguerite, et al. (2006). "Sequoia 2.10 User's Guide." Retrieved 2008 Mayo, from http://sequoia.continuent.org/doc/latest/userGuide/sequoia-user-guide.pdf.
8. Cecchet, E., J. Marguerite, et al. (2003). "RAIDb: Redundant Array of Inexpensive Databases." Retrieved 2008 mayo, from http://c-jdbc.objectweb.org/current/doc/RR-C-JDBC.pdf.
9. Continuent.org. (2007a). "Myosotis " Retrieved 2008 Mayo, from http://myosotis.continuent.org/.
10. Continuent.org. (2007b). "Sequoia 3.0 Basic Concepts." Retrieved 2008 Abril, from http://sequoia.continuent.org/doc/latest/sequoia-basic-concepts.pdf.
11. Culebro Juárez, M., W. G. Gómez Herrera, et al. (2006). Software libre vs software propietario. Ventajas y desventajas. México.
12. Galstad, E. (2007). "Nagios Version 3.x Documentation." Retrieved 2008 Junio, from http://www.nagios.org.
13. Gutiérrez Díaz, D., A. González Pérez, et al. (2003). "El cluster Beowulf del centro nacional de bioinformática: diseño, montaje y evaluación preliminar." Retrieved 2008 Febrero, from http://revistas.mes.edu.cu:9900/EDUNIV/03-Revistas-Cientificas/Ingenieria-Industrial/2003/3/10403302.pdf.
14. Horman, S. (2003). "LInux Virtual Server tutorial." Retrieved 2008 Marzo from http://www.ultramonkey.org.
15. Kneschke, J. (2007). "MySQL Proxy." Retrieved 2008 Febrero, from http://forge.mysql.com/w/images/d/dd/MySQL-Proxy.pdf.
16. Linux-HA. (2006). "Heartbeat." Retrieved 2008 Mayo, from http://www.linux-ha.org/Heartbeat.

17. LVS. (1998). "Job Scheduling Algorithms in Linux Virtual Server." Retrieved 2008 Marzo, from http://www.linuxvirtualserver.org/docs/scheduling.html.
18. Martínez Jiménez, M. and G. Bergés Pujol (2003). Arquitecturas de Clustering de Alta Disponibilidad y Escalabilidad (Linux Virtual Server), ACADE (LVS). Proyecto Final de Carrera.
19. Maxia, G. (2007). "Getting Started with MySQL Proxy." Retrieved 2008 Febrero, from http://dev.mysql.com/tech-resources/articles/proxy-gettingstarted.html.
20. Meier, J. D., C. Farre, et al. (2007). Performance Testing Guidance for Web Applications. Microsoft, Microsoft**:** 221.
21. Millán Tejedor, R. J. "SNMPv3 (Simple Network Management Protocol version 3)." Retrieved 2008 Junio, from http://www.ramonmillan.com/documentos/snmpv3.pdf.
22. Mouse, D. (2006). "SQL Relay web site." Retrieved 2008 Febrero, from http://sqlrelay.sourceforge.net/.
23. MySQL AB (2008a). Chapter 13. Storage Engines. MySQL 5.0 Reference Manual.
24. MySQL AB (2008b). Chapter 16. Replication. MySQL 5.0 Reference Manual.
25. MySQL AB (2008c). Chapter 17. MySQL Cluster. MySQL 5.0 Reference Manual.
26. MySQL AB (2008d). Chapter 26. MySQL Proxy. MySQL 5.0 Reference Manual.
27. Oetiker, T. (2008a). "How to use RRDtool with MRTG." Retrieved 2008 Mayo, from http://oss.oetiker.ch/mrtg/doc/mrtg-rrd.en.html.
28. Oetiker, T. (2008b). "What is MRTG?" Retrieved 2008 Mayo, from http://oss.oetiker.ch/mrtg/doc/mrtg.en.html.
29. Oliveira, R. (2006). "GORDA: An open architecture for database replication." Retrieved 2008 Abril, from http://gorda.di.uminho.pt.
30. Pressman, R. S. (2001). Ingeniería del Software. Un enfoque práctico, Mc. Graw Hill (España).
31. Robertson, A. (2000). Linux − HA Heartbeat System Design. 4th Annual Linux Showcase & Conference, Atlanta, Atlanta, Georgia, USA, The USENIX Association.
32. Softel (2006). Arquitectura de Software para los componentes a emplear por el Sistema de Información para la Salud. Procedimientos. Softel. Ciudad Habana., Softel. 2007: 13.
33. Softel (2007). Arquitectura, normas y tecnologías para el desarrollo de aplicaciones informáticas para la Salud Pública en Cuba. Procedimientos. Softel. Ciudad Habana. 2007: 14.

BIBLIOGRAFÍA

1. Aguilar Roselló, V.J., *Clustering de alta disponibilidad bajo GNU / LINUX.*, in *Departamento de Lenguajes y Sistemas Informáticos.* 2001, Universidad de Alicante.
2. Andreasson, O. *Iptables tutorial 1.2.2.* 2007 [cited 2008 Junio]; Available from: http://www.netfilter.org/.
3. Apache Jakarta Project. *JMeter User's Manual*. 2008 [cited 2008 Mayo]; Available from: http://jakarta.apache.org/jmeter/usermanual/index.html.
4. Balling, D.J. and J. Zawodny, *High performance mySQL.* 1rst ed. 2004: O'Reilly p. 294.
5. Barroso, L.A., J. Dean, and U. Hölzle. *Web search for a planet: The Google cluster architecture*. 2003 [cited 2008 Marzo]; Available from: http://labs.google.com/papers/googlecluster-ieee.pdf.
6. BLSF Equipo de desarrollo, *Chapter 11. Utilidades del Sistema. Sysstat-5.1.5*, in *Más Allá de Linux*. 2005.
7. Bourke, T., *Server Load Balancing*. 1st ed. 2001, Sebastopol: O'Reilly & Associates, Inc. 182.
8. Cecchet, E., et al. *Sequoia 2.10 User's Guide*. 2006 [cited 2008 Mayo]; Available from: http://sequoia.continuent.org/doc/latest/userGuide/sequoia-user-guide.pdf.
9. Cecchet, E., J. Marguerite, and W. Zwaenepoel. *RAIDb: Redundant Array of Inexpensive Databases.* 2003 [cited 2008 mayo]; Available from: http://c-jdbc.objectweb.org/current/doc/RR-C-JDBC.pdf.
10. Cecchet, E.M., Julie; Zwaenepoel, Willy. *C-JDBC, Flexible database clustering midleware.* 2006 [cited 2008 mayo]; Available from: http://c-jdbc.objectweb.org/current/doc/C-JDBC_Flexible_Database_Clustering_Middleware.pdf.
11. Continuent.org. *Sequoia 3.0 Basic Concepts*. 2007 [cited 2008 Abril]; Available from: http://sequoia.continuent.org/doc/latest/sequoia-basic-concepts.pdf.
12. Continuent.org. *Myosotis* 2007 [cited 2008 Mayo]; Available from: http://myosotis.continuent.org/.
13. Culebro Juárez, M., W.G. Gómez Herrera, and S. Torres Sánchez, *Software libre vs software propietario. Ventajas y desventajas*. 2006: México.
14. Galstad, E. *Nagios Version 3.x Documentation*. 2007 [cited 2008 Junio]; Available from: http://www.nagios.org.
15. Gutiérrez Díaz, D., A. González Pérez, and J.P. Febles Rodríguez. *El cluster Beowulf del centro nacional de bioinformática: diseño, montaje y evaluación preliminar*. 2003 [cited 2008 Febrero]; Available from: http://revistas.mes.edu.cu:9900/EDUNIV/03-Revistas-Cientificas/Ingenieria-Industrial/2003/3/10403302.pdf.
16. Horman, S. *LInux Virtual Server tutorial.* 2003 [cited 2008 Marzo]; Available from:

http://www.ultramonkey.org.

17. Kneschke, J. *MySQL Proxy.* 2007 [cited 2008 Febrero]; Available from: http://forge.mysql.com/w/images/d/dd/MySQL-Proxy.pdf.
18. Kopper, K., *The Linux Enterprise Cluster.* 2005, San Francisco: No Starch Press, Inc. 464.
19. Linux-HA. *Getting Started with Linux-HA (Heartbeat)*. [cited 2008 Mayo]; Available from: http://www.linux-ha.org/GettingStarted.
20. LVS. *Job Scheduling Algorithms in Linux Virtual Server.* 1998 [cited 2008 Marzo]; Available from: http://www.linuxvirtualserver.org/docs/scheduling.html.
21. Martínez Jiménez, M. and G. Bergés Pujol, *Arquitecturas de Clustering de Alta Disponibilidad y Escalabilidad (Linux Virtual Server), ACADE (LVS).* 2003.
22. Maxia, G. *Getting Started with MySQL Proxy*. 2007 [cited 2008 Febrero]; Available from: http://dev.mysql.com/tech-resources/articles/proxy-gettingstarted.html.
23. Meier, J.D., et al., *Performance Testing Guidance for Web Applications.*, Microsoft, Editor. 2007, Microsoft. p. 221.
24. Millán Tejedor, R.J. *SNMPv3 (Simple Network Management Protocol version 3)*. [cited 2008 Junio]; Available from: http://www.ramonmillan.com/documentos/snmpv3.pdf.
25. Mouse, D. *SQL Relay web site.* 2006 [cited 2008 Febrero]; Available from: http://sqlrelay.sourceforge.net/.
26. MySQL AB. *MySQL 5.1 Reference Manual.* 2006 [cited 2008 Abril]; Available from: http://www.mysql.com.
27. Oetiker, T. *What is MRTG?* 2008 [cited 2008 Mayo]; Available from: http://oss.oetiker.ch/mrtg/doc/mrtg.en.html.
28. Oetiker, T. *How to use RRDtool with MRTG*. 2008 [cited 2008 Mayo]; Available from: http://oss.oetiker.ch/mrtg/doc/mrtg-rrd.en.html.
29. Oliveira, R. *GORDA: An open architecture for database replication.* 2006 [cited 2008 Abril]; Available from: http://gorda.di.uminho.pt.
30. Paredes, J.P. *Alta disponibilidad para Linux.* 2005 [cited 2008 Junio]; Available from: http://www.ibiblio.org/pub/Linux/docs/LuCaS/Presentaciones/200103hispalinux/paredes/pdf/LinuxHA.pdf.
31. Pinheiro, E., et al., *Load Balancing and Unbalancing for Power and Performance in Cluster-Based Systems.*, R.U. Department of Computer Science, Editor, Rutgers University: Piscataway. p. 11.
32. Pressman, R.S., *Ingeniería del Software. Un enfoque práctico*. 5th ed. 2001: Mc. Graw Hill (España).
33. Pujol García, J.C., et al. *Altas prestaciones de servidores con aplicaciones de salud.* in *Uciencia*

2007. 2007. Ciudad Habana: UCI.

34. Robertson, A. *Linux − HA Heartbeat System Design.* in *4th Annual Linux Showcase & Conference, Atlanta*. 2000. Atlanta, Georgia, USA: The USENIX Association.
35. Softel, *Arquitectura de Software para los componentes a emplear por el Sistema de Información para la Salud.*, in *Procedimientos*, Softel, Editor. 2006, Softel: Ciudad Habana. p. 13.
36. Softel, *Arquitectura, normas y tecnologías para el desarrollo de aplicaciones informáticas para la Salud Pública en Cuba.*, in *Procedimientos*, Softel, Editor. 2007: Ciudad Habana. p. 14.

ANEXOS

Anexo 1. Documentos de diseño de las soluciones propuestas

Listado de los documentos de diseño de las soluciones propuestas. Estos documentos se pueden encontrar en formato digital en el CD-ROM adjunto a este informe.

Instalación y configuración de la solución LVS y replicación MySQL para RUS

Nombre del archivo: Instalación y configuración de la solución LVS y replicación MySQL para RUS.pdf

Ubicación: Soluciones\LVS y replicación MySQL para RUS\

Ubicación de los ficheros de configuración adjuntos: Soluciones\LVS y replicación MySQL para RUS\ficheros de configuración\

Instalación y configuración de la solución Myosotis y Sequoia para RUS

Nombre del archivo: Instalación y configuración de la solución Myosotis y Sequoia para RUS.pdf

Ubicación: Soluciones\Myosotis y Sequoia para RUS\

Ubicación de los ficheros de configuración adjuntos: Soluciones\Myosotis y Sequoia para RUS\ficheros de configuración\

Instalación y configuración de la solución Sequoia para RCD

Nombre del archivo: Instalación y configuración de la solución Sequoia para RCD.pdf

Ubicación: Soluciones\Sequoia para RCD\

Ubicación de los ficheros de configuración adjuntos: Soluciones\Sequoia para RCD\ficheros de configuración\

Instalación y configuración de la solución Myosotis y Sequoia para BM

Nombre del archivo: Instalación y configuración de la solución Myosotis y Sequoia para BM.pdf

Ubicación: Soluciones\Myosotis y Sequoia para BM\

Ubicación de los ficheros de configuración adjuntos: Soluciones\Myosotis y Sequoia para BM\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_[AR]

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: RUS para la aplicación Registro de Unidades de Salud, BM para la aplicación Balance Material y RCD para la aplicación Registro centralizado de donantes.

El campo “capa de la aplicación” se utiliza para especificar la capa donde se despliega el cluster, capa de datos (CD).

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.

El campo opcional “AR” se utiliza cuando el archivo contiene el análisis de resultados de la prueba.

Anexo 3. Documentos de planificación y análisis de resultados de las pruebas

Listado de documentos de planificación y análisis de resultados de las pruebas realizadas durante la investigación. Estos documentos se pueden encontrar en formato digital en el CD-ROM adjunto a este informe.

Prueba RUS-CD-CO-01

Nombre del archivo de planificación de la prueba: RUS-CD-CO-01.pdf

Nombre del archivo de análisis de resultados de la prueba: RUS-CD-CO-01_AR.pdf

Ubicación de los archivos: Pruebas\Pruebas a RUS\LVS y replicación MySQL\1ra iteración\

Prueba RUS-CD-CO-02

Nombre del archivo de planificación de la prueba: RUS-CD-CO-02.pdf

Nombre del archivo de análisis de resultados de la prueba: RUS-CD-CO-02_AR.pdf

Ubicación de los archivos: Pruebas\Pruebas a RUS\LVS y replicación MySQL\1ra iteración\

Prueba RUS-CD-CO-03

Nombre del archivo de planificación de la prueba: RUS-CD-CO-03.pdf

Nombre del archivo de análisis de resultados de la prueba: RUS-CD-CO-03_AR.pdf

Ubicación de los archivos: Pruebas\Pruebas a RUS\LVS y replicación MySQL\1ra iteración\

Prueba RUS-CD-CO-04

Nombre del archivo de planificación de la prueba: RUS-CD-CO-04.pdf

Nombre del archivo de análisis de resultados de la prueba: RUS-CD-CO-04_AR.pdf

Ubicación de los archivos: Pruebas\Pruebas a RUS\LVS y replicación MySQL\1ra iteración\

Prueba RUS-CD-CO-05

Nombre del archivo de planificación de la prueba: RUS-CD-CO-05.pdf

Nombre del archivo de análisis de resultados de la prueba: RUS-CD-CO-05_AR.pdf

Ubicación de los archivos: Pruebas\Pruebas a RUS\LVS y replicación MySQL\1ra iteración\

Prueba RUS-CD-CO-06

Nombre del archivo de planificación de la prueba: RUS-CD-CO-06.pdf

Nombre del archivo de análisis de resultados de la prueba: RUS-CD-CO-06_AR.pdf

Ubicación de los archivos: Pruebas\Pruebas a RUS\LVS y replicación MySQL\2da iteración\

Prueba RUS-CD-CO-07

Nombre del archivo de planificación de la prueba: RUS-CD-CO-07.pdf

Nombre del archivo de análisis de resultados de la prueba: RUS-CD-CO-07_AR.pdf

Ubicación de los archivos: Pruebas\Pruebas a RUS\LVS y replicación MySQL\2da iteración\

Prueba RUS-CD-CO-08

Nombre del archivo de planificación de la prueba: RUS-CD-CO-08.pdf

Nombre del archivo de análisis de resultados de la prueba: RUS-CD-CO-08_AR.pdf

Ubicación de los archivos: Pruebas\Pruebas a RUS\LVS y replicación MySQL\2da iteración\

Prueba RUS-CD-CO-09

Nombre del archivo de planificación de la prueba: RUS-CD-CO-09.pdf

Nombre del archivo de análisis de resultados de la prueba: RUS-CD-CO-09_AR.pdf

Ubicación de los archivos: Pruebas\Pruebas a RUS\LVS y replicación MySQL\2da iteración\

Prueba RUS-CD-CO-10

Nombre del archivo de planificación de la prueba: RUS-CD-CO-10.pdf

Nombre del archivo de análisis de resultados de la prueba: RUS-CD-CO-10_AR.pdf

Ubicación de los archivos: Pruebas\Pruebas a RUS\LVS y replicación MySQL\2da iteración\

Prueba RUS-CD-CA-11

Nombre del archivo de planificación de la prueba: RUS-CD-CA-11.pdf

Nombre del archivo de análisis de resultados de la prueba: RUS-CD-CA-11_AR.pdf

Ubicación de los archivos: Pruebas\Pruebas a RUS\ LVS y replicación MySQL \

Prueba RUS-CD-CA-12

Nombre del archivo de planificación de la prueba: RUS-CD-CA-12.pdf

Nombre del archivo de análisis de resultados de la prueba: RUS-CD-CA-12_AR.pdf

Ubicación de los archivos: Pruebas\Pruebas a RUS\ LVS y replicación MySQL \

Prueba RUS-CD-CO-13

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

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

Ubicación de los archivos: Pruebas\Pruebas a RUS\Myosotis y Sequoia\

Prueba RUS-CD-CO-14

Nombre del archivo de planificación de la prueba: RUS-CD-CO-14.pdf

Nombre del archivo de análisis de resultados de la prueba: RUS-CD-CO-14_AR.pdf

Ubicación de los archivos: Pruebas\Pruebas a RUS\Myosotis y Sequoia\

Prueba RUS-CD-CO-15

Nombre del archivo de planificación de la prueba: RUS-CD-CO-15.pdf

Nombre del archivo de análisis de resultados de la prueba: RUS-CD-CO-15_AR.pdf

Ubicación de los archivos: Pruebas\Pruebas a RUS\Myosotis y Sequoia\

Prueba RUS-CD-CO-16

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

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

Ubicación de los archivos: Pruebas\Pruebas a RUS\Myosotis y Sequoia\

Prueba RUS-CD-CA-17

Nombre del archivo de planificación de la prueba: RUS-CD-CA-17.pdf

Nombre del archivo de análisis de resultados de la prueba: RUS-CD-CA-17_AR.pdf

Ubicación de los archivos: Pruebas\Pruebas a RUS\Myosotis y Sequoia\

Prueba RUS-CD-CA-18

Nombre del archivo de planificación de la prueba: RUS-CD-CA-18.pdf

Nombre del archivo de análisis de resultados de la prueba: RUS-CD-CA-18_AR.pdf

Ubicación de los archivos: Pruebas\Pruebas a RUS\Myosotis y Sequoia\

Prueba RCD-CD-CO-19

Nombre del archivo de planificación de la prueba: RCD-CD-CO-19.pdf

Nombre del archivo de análisis de resultados de la prueba: RCD-CD-CO-19_AR.pdf

Ubicación de los archivos: Pruebas\Pruebas a RCD\

Prueba RCD-CD-CO-20

Nombre del archivo de planificación de la prueba: RCD -CD-CO-20.pdf

Nombre del archivo de análisis de resultados de la prueba: RCD-CD-CO-20_AR.pdf

Ubicación de los archivos: Pruebas\Pruebas a RCD\

Prueba RCD-CD-CO-21

Nombre del archivo de planificación de la prueba: RCD -CD-CO-21.pdf
Nombre del archivo de análisis de resultados de la prueba: RCD-CD-CO-21_AR.pdf
Ubicación de los archivos: Pruebas\Pruebas a RCD\

Prueba RCD-CD-CO-22

Nombre del archivo de planificación de la prueba: RCD -CD-CO-22.pdf
Nombre del archivo de análisis de resultados de la prueba: RCD-CD-CO-22_AR.pdf
Ubicación de los archivos: Pruebas\Pruebas a RCD\

Prueba RCD-CD-CA-23

Nombre del archivo de planificación de la prueba: RCD -CD-CA-23.pdf
Nombre del archivo de análisis de resultados de la prueba: RCD-CD-CA-23_AR.pdf
Ubicación de los archivos: Pruebas\Pruebas a RCD\

Prueba RCD-CD-CA-24

Nombre del archivo de planificación de la prueba: RCD -CD-CA-24.pdf
Nombre del archivo de análisis de resultados de la prueba: RCD-CD-CA-24_AR.pdf
Ubicación de los archivos: Pruebas\Pruebas a RCD\

Prueba BM-CD-CO-25

Nombre del archivo de planificación de la prueba: BM -CD-CO-25.pdf
Nombre del archivo de análisis de resultados de la prueba: BM-CD-CO-25_AR.pdf
Ubicación de los archivos: Pruebas\Pruebas a BM\1ra iteración\

Prueba BM-CD-CO-26

Nombre del archivo de planificación de la prueba: BM -CD-CO-26.pdf
Nombre del archivo de análisis de resultados de la prueba: BM-CD-CO-26_AR.pdf
Ubicación de los archivos: Pruebas\Pruebas a BM\1ra iteración\

Prueba BM-CD-CO-27

Nombre del archivo de planificación de la prueba: BM -CD-CO-27.pdf
Nombre del archivo de análisis de resultados de la prueba: BM-CD-CO-27_AR.pdf
Ubicación de los archivos: Pruebas\Pruebas a BM\1ra iteración\

Prueba BM-CD-CO-28

Nombre del archivo de planificación de la prueba: BM -CD-CO-28.pdf
Nombre del archivo de análisis de resultados de la prueba: BM-CD-CO-28_AR.pdf
Ubicación de los archivos: Pruebas\Pruebas a BM\1ra iteración\

Prueba BM-CD-CO-29

Nombre del archivo de planificación de la prueba: BM -CD-CO-29.pdf
Nombre del archivo de análisis de resultados de la prueba: BM-CD-CO-29_AR.pdf
Ubicación de los archivos: Pruebas\Pruebas a BM\2da iteración\

Prueba BM-CD-CO-30

Nombre del archivo de planificación de la prueba: BM -CD-CO-30.pdf
Nombre del archivo de análisis de resultados de la prueba: BM-CD-CO-30_AR.pdf
Ubicación de los archivos: Pruebas\Pruebas a BM\2da iteración\

Prueba BM-CD-CO-31

Nombre del archivo de planificación de la prueba: BM -CD-CO-31.pdf
Nombre del archivo de análisis de resultados de la prueba: BM-CD-CO-31_AR.pdf
Ubicación de los archivos: Pruebas\Pruebas a BM\2da iteración\

Prueba BM-CD-CO-32

Nombre del archivo de planificación de la prueba: BM -CD-CO-32.pdf
Nombre del archivo de análisis de resultados de la prueba: BM-CD-CO-32_AR.pdf
Ubicación de los archivos: Pruebas\Pruebas a BM\2da iteración\

GLOSARIO DE TÉRMINOS

API: (*Application Programming Interface* / Interfaz de Programación de Aplicaciones). Conjunto de funciones, procedimientos o métodos ofrecidos por determinada biblioteca para ser utilizado por otro *software* como una capa de abstracción. Interfaz de comunicación entre componentes de *software*.

Bug: Mal funcionamiento de un elemento de *software*.

Cuello de botella (*Bottleneck*): es el embotellamiento de paquetes de datos (información) que circulan por una conexión; causa demoras en la comunicación.

DBMS: (*Database Management System* / Sistema Gestor de Bases de datos). Permite la administración de los datos físicos a través de estructuras lógicas que establecen vínculos de integridad, métodos de acceso y organización entre ellos.

Demonio: En Unix/Linux se conoce como un programa que permanece en segundo plano ejecutándose continuamente para brindar algún tipo de servicio.

Granja de servidores: Agrupación de servidores que brindan determinados servicios a un sistema en común.

Java: Lenguaje de programación orientado a objetos y multiplataforma basado en la sintaxis del lenguaje C y C++. La compilación del código java se realiza en un código intermedio interpretado por una máquina virtual.

JDBC: Es el acrónimo de *Java Database Connectivity*, un API que permite la ejecución de operaciones sobre bases de datos desde el lenguaje de programación Java, independientemente del sistema operativo donde se ejecute o de la base de datos a la cual se accede. Las operaciones realizadas sobre las bases de datos se especifican haciendo uso de un lenguaje de consulta estructurado (*SQL*).

Master: En la replicación de bases de datos el servidor *Master* (maestro o amo) es donde se realizan todas las consultas de modificación de datos las cuales se almacenan en el *log* binario del servidor.

Middleware: Componente de *software* que actúa como intermediario entre otros componentes de *software*, generalmente en el marco de la interacción cliente/servidor.

MySQL: Sistema gestor de bases de datos relacionales (*Relational Database Management System* / RDMS), multihilo y multiusuario que implementa múltiples motores de almacenamiento (*storage engines*). Puede ser utilizado desde varias plataformas.

Nodo: En la terminología de cluster un nodo es cualquier máquina física que juega un determinado papel dentro del cluster (balanceador de carga, máquina de respaldo, servidor real, etc).

PHP (*Hypertext Pre-processor*): Lenguaje de programación interpretado ampliamente utilizado y diseñado para el desarrollo web. Se ejecuta del lado del servidor y tiene soporte para varias bases de datos (MySQL, Oracle, Sybase, PostgreSQL, ODBC, etc.). Es de código abierto y multiplataforma.

Pool de conexiones: Grupo de conexiones que se establecen a una base de datos y permanecen abiertas para que puedan ser reutilizadas en la realización de múltiples consultas o actualizaciones. Generalmente estas conexiones permanecen abiertas y sólo se cierran una vez que termine la ejecución del programa que las originó.

RDBMS: (*Relational Database Management System* / Sistema Gestor de Bases de datos Relacionales). Constituyen el siguiente paso en la evolución de los DBMS. El almacenamiento de los datos se realiza por medio de una estructura tabular llamada relación o tabla que está compuesta por filas y columnas a las cuales se accede a través de un lenguaje de alto nivel.

Rendimiento: El rendimiento de un servidor de bases de datos puede ser medido a través del consumo que realiza de sus recursos, tales como: CPU (*Central Processing Unit* / Unidad Central de Procesamiento), RAM (*Random Access Memory* / Memoria de Acceso Aleatorio), tráfico de red, etc. El rendimiento de un sistema puede medirse a través de los tiempos de respuesta de las consultas realizadas a los servidores de bases de datos.

Script: Fichero que contiene un conjunto de instrucciones que permiten la automatización de determinadas tareas y son ejecutados por el *shell* de un sistema operativo. Muy utilizados en la administración de sistemas Unix/Linux.

Servidor: Un servidor es un *software* que se ejecuta continuamente en una máquina computadora, manteniéndose a la espera de peticiones de ejecución que le hará un cliente o un usuario. A menudo se le llama incorrectamente servidor al ordenador físico en el cual funciona este *software*.

Slave: En la replicación de bases de datos los servidores *Slave* (esclavo) leen las consultas almacenadas en el *log* binario del servidor *Master* y las ejecutan localmente, de esta manera mantienen una copia exacta de los datos almacenados en el *Master*.

Storage engine: Componente de un DBMS utilizado para crear, devolver, actualizar y borrar datos de una base de datos. Para MySQL 5 algunos de los tipos de *storage engine* son: MyISAM, InnoDB, MERGE, MEMORY, BDB, EXAMPLE, FEDERATED, ARCHIVE, CSV, BLACKHOLE. (MySQL AB 2008a)