Universidad de las Ciencias Informáticas
Facultad 15

Clúster de altas prestaciones para medianas y pequeñas bases de datos que utilizan a PostgreSQL como sistema de gestión de bases de datos.

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas.

Autor: Yoemir Orduñez Santana

Tutor: Ing. Adrian Misael Peña Montero

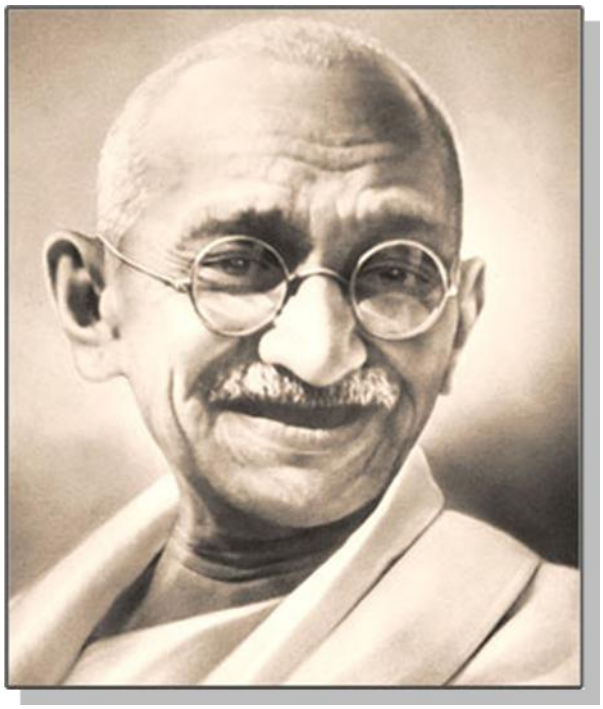
Co-Tutor: Ing. Marcos Luis Ortiz Valmaseda

Asesores: Ing. Daymel Bonne Solís

Ing. Persy Morell Guerra

Ciudad de la Habana, Junio 2010

"Año 52 de la Revolución"



“Nuestra recompensa se encuentra en el esfuerzo y no en el resultado. Un esfuerzo total es una victoria completa”.

Mahatma Gandhi

DECLARACIÓN DE AUTORÍA

Declaro ser autor de la presente tesis y reconozco a la Universidad de las Ciencias Informáticas, y al Centro de Tecnología de Almacenamiento y Análisis de Datos los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

Yoemir Orduñez Santana

_____

Firma del Autor

Adrián Misael Peña Montero	Marcos L. Ortíz Valmaseda	Daymel Bonne Solís
_____	_____	_____
Firma del Tutor	Firma del Tutor	Firma del Tutor

Persy Morell Guerra

_____

Firma del Tutor

Ing. Adrián Misael Peña: Ingeniero en Ciencias Informáticas.

Centro de desarrollo: **DATEC**

Dirección: Universidad de las Ciencias Informáticas

E-mail: ampena@uci.cu

Ing. Marcos Luis Ortíz Valmaseda: Ingeniero en Ciencias Informáticas.

Centro de desarrollo: **DATEC**

Dirección: Universidad de las Ciencias Informáticas

E-mail: mlortiz@uci.cu

Ing. Daymel Bonne Solís: Ingeniero en Ciencias Informáticas.

Centro de desarrollo: **DATEC**

Dirección: Universidad de las Ciencias Informáticas

E-mail: dbonne@uci.cu

Tte. Persy Morell Guerra: Ingeniero en Ciencias Informáticas.

Centro de desarrollo: **UCIFAR**

Dirección: Universidad de las Ciencias Informáticas

E-mail: pmorell@uci.cu

AGRADECIMIENTOS

A mi madre querida, por haber hecho de mi la persona que soy, por haber logrado tantas cosas gracias a ella, por ser faro y guía en mi vida, por ser mi fuente de inspiración, por ser la persona en la que mas confío, por ser mi ¡todo! en la vida.

A mi querida abuela, que no tuvo la posibilidad de verme graduado, pero que siempre albergó la confianza de ver en mí, un universitario.

A mi padre por los consejos y apoyo que siempre me ha brindado, además de la confianza que ha depositado en mí.

A mi hermano José Raúl por confiar en mí.

A mis queridos tíos Miguel y Joseito por ser como padres para mi, por el inmenso cariño que nos profesamos, por las tantas cosas que hemos compartido.

A los tutores de esta investigación, por el duro trabajo que hicieron conmigo, en especial a Adrian por ser guía de la tesis, a Marcos y Bonne por las constantes preguntas, en general a los tres por preocuparse tanto por este trabajo, por dejarme ser una polilla y filtrarme entre sus conocimientos.

A mi primo Miguelito, por ser siempre tan perfeccionista, por darme consejos constantemente, por su afán de triunfar en la vida.

A mis primas(o), que de verdad que se han portado como hermanas(o) conmigo, no tengo quejas de ellos, siempre estuvieron “ahí” para mí.

Finalizando pero para nada menos importante, al contrario, pilar que influyó en mi carrera, en mi graduación, hizo que mejorara mi formación como persona, mis agradecimientos a mi novia Beatriz, por aguantarme durante estos 2 años y medios de relación, por estar conmigo en los malos y buenos momentos por los que he pasado.

A mis dos amigos Ariel y Marcel, por su incondicional amistad.

A todos mis amigos que de una forma u otra han influido en mi desempeño como estudiante.

DEDICATORIA

A los profes que influyeron en mi formación profesional.

A la memoria de mi abuela.

A la memoria de mi padre.

Muy en especial a mi mamá.

A mi papá Raúl.

A mis hermanos.

A mis tíos.

A mis primos.

A mis amigos.

A toda mi familia en general.

Resumen

Cuba está inmersa en el proceso de informatización de la mayoría de los sectores de la sociedad, por lo que adjunto a esto, han surgido un grupo de aplicaciones y sistemas para prestar servicios a dicho objetivo.

Muchas de estas aplicaciones se implementan en la Universidad de las Ciencias Informáticas, más conocida como la universidad de las Tecnologías, contribuyendo con la misma varias entidades del país. En su gran mayoría la implementación de estos sistemas se realiza utilizando *software* libre, siendo su explotación de forma centralizada.

La meta y principal objetivo es implementar una solución que aumente la capacidad de respuesta y disponibilidad en los servidores donde se despliegan medianas y pequeñas bases de datos, utilizando el *SGBD* PostgreSQL, mediante la implementación de un clúster de alto rendimiento y alta disponibilidad.

La solución propuesta contribuyó a un aumento de la disponibilidad y una disminución de los tiempos de respuestas en un cierto grupo de aplicaciones, disminuyendo así el consumo de recursos ejercido sobre los servidores de bases de datos.

Palabras claves

Clúster, capacidad de respuesta, clúster de alto rendimiento, clúster de alta disponibilidad.

Introducción.

El proceso de informatización en que está actualmente inmersa Cuba, ha desencadenado el desarrollo de un gran número de aplicaciones para automatizar varios de los sectores de la sociedad. Los sistemas a desarrollar son implementados en su mayoría en la Universidad de las Ciencias Informáticas, colaborando con la misma varias entidades del país.

Siguiendo las políticas que se han dictado sobre el uso de *software* libre y la utilización del mismo para el desarrollo de dichas aplicaciones, la mayoría de éstas utilizan el sistema operativo *Linux*, desplegándose sobre servidores que utilizan dicho sistema operativo algunas de sus distribuciones.

La explotación de forma centralizada de estas aplicaciones, conlleva a que muchos usuarios accedan de manera concurrente, por lo que se generan grandes volúmenes de carga para los servidores de bases de datos sobre los que se ejecutan, por lo que se precisa de un *hardware* con altas capacidades de procesamiento.

Debido a la generalidad en el diseño de las aplicaciones y como consecuencia de su explotación central, la capacidad de procesamiento de los servidores de bases de datos sobre los que se ejecutan varias de estas, es inferior a la requerida y aunque el *hardware* puede ser mejorado, existe un límite.

Brindar servicios utilizando servidores con capacidad de procesamiento inferior a la que se requiere por las aplicaciones, afecta el rendimiento y correcto funcionamiento del sistema; en estas condiciones dicho sistema puede alcanzar tiempos de respuestas inaceptables para los usuarios que acceden a la aplicación o simplemente éstas quedan fuera de servicio por una determinada sobrecarga.

Esta investigación se realiza para dar solución al problema de aumentar la capacidad de respuesta y la disponibilidad en los servidores de bases de datos PostgreSQL.

Planteándose el anterior problema, el **objetivo de la investigación** es diseñar una propuesta de solución, utilizando herramientas libres, que permita la obtención de mayor

capacidad de respuesta y disponibilidad en los servidores donde se despliegan medianas y pequeñas bases de datos que utilizan el sistema de gestión de bases de datos PostgreSQL.

Para lograr el objetivo de dicha investigación se planteó la siguiente **idea a defender**: la sustitución de un servidor PostgreSQL por un clúster para este *SGBD,* permitirá obtener una mayor disponibilidad y capacidad de respuesta en sistemas que utilicen medianas y pequeñas bases de datos.

El **objeto de estudio** para la investigación en curso es: los servidores de bases de datos PostgreSQL donde se despliegan medianas y pequeñas bases de datos.

El **campo de acción** del anterior objetivo es: la capacidad de respuesta y la disponibilidad de servidores de bases de datos PostgreSQL donde se despliegan medianas y pequeñas bases de datos.

Para dar cumplimiento a los anteriores objetivos se han definido un grupo de tareas que encaminarán el transcurso de la investigación:

Realizar un estudio de las técnicas para el desarrollo de clúster de base de datos y del *software* que puede ser usado para su implementación utilizando el sistema de gestión de bases de datos PostgreSQL.

1. Estudio de las características de las aplicaciones que serán ejecutadas en el clúster.
2. Diseñar una o varias propuestas de solución, basadas en técnicas de clúster de base de datos para el sistema de gestión de bases de datos PostgreSQL, que permitan obtener mayor capacidad de respuesta y disponibilidad en los servidores donde se despliegan medianas y pequeñas bases de datos.
3. Realizar pruebas de funcionalidad y carga que permitan validar la o las propuestas de solución que se diseñen.
4. Redactar los manuales necesarios para el montaje y mantenimiento de la o las soluciones que se diseñen.

Como resultado del presente trabajo se espera obtener un diseño de solución, que permita aumentar la capacidad de respuesta y la disponibilidad en los servidores de bases de datos PostgreSQL.

La estructura del documento consta de tres capítulos, cuyo contenido se muestra a continuación:

En el Capítulo 1 se abordan conceptos generales sobre la tecnología clúster, rasgos fundamentales del sistema de gestión de bases de datos PostgreSQL, las herramientas que se utilizan para balancear las cargas sobre dicho *SGBD*, las herramientas para implementar las alta disponibilidad en el clúster, la descripción de los tipos de pruebas que se pueden realizar y de las que se van a aplicar para medir el correcto funcionamiento y rendimiento de la aplicación.

En el Capítulo 2 se describe el diseño lógico y físico de la solución además del diseño de interconexión entre los nodos, temas referentes al mantenimiento del mismo así como su seguridad.

En el Capítulo 3 se describe todo el procedimiento de las pruebas de configuración, las funcionalidades del sistema a probar, así como las pruebas que se le aplican a la propuesta de diseño genérico de la solución.

Capítulo 1. Fundamentación Teórica

Introducción a la tecnología clúster.

Actualmente el volumen de información que se maneja entre equipos que realizan operaciones de cómputo, crece a gran escala, convirtiéndose esto en tareas muy complejas, por lo que se requiere de un sistema con condiciones mejoradas de *hardware* que posibilite aumentar el manejo de dichos flujos de información, esta tarea en una gran mayoría de aplicaciones, desde una simple computadora no se lograría.

Cuando muchos usuarios acceden de manera concurrente a una aplicación, ésta necesita ser capaz de procesar todas las peticiones y dar respuestas en tiempos aceptables para el usuario final. Un sistema como el que se describió anteriormente está limitado por los recursos de *hardware* , lo que trae consigo un fuerte problema, ya que con el paso del tiempo el acceso de los usuarios de manera concurrente asciende grandemente y los recursos son insuficientes, como solución se puede aumentar la capacidad de procesamiento con nuevos componentes de *hardware* o adquirir una nueva máquina con mejores condiciones , pero esto tiene un alto valor que incurre como un costo adicional al de *software* y los propios del mantenimiento del *hardware*.

Existen otras formas de mejorar el rendimiento de los servidores, entre ellas se encuentra la implementación de un clúster de servidores, ésta constituye una solución económicamente más rentable que el uso de una supercomputadora y brinda las mismas funcionalidades con una mayor flexibilidad.

1.1 Características generales de la tecnología clúster.

1.1.1 Definición

Un clúster 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. (Almaguer, y otros, 2008)

Ventajas.

La implementación de un clúster brinda gran flexibilidad dada las características de sus nodos. Los nodos de un clúster pueden tener las mismas configuraciones de *hardware* y sistema operativo, cuando esto ocurre se le llama clúster 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 clúster se denomina clúster semi-homogéneo. Como última alternativa cuando los nodos tienen diferente *hardware* y sistema operativo se le llama clúster heterogéneo. La flexibilidad de un clúster hace más fácil y económica su construcción.

Cada uno de los nodos de un clúster 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 clúster 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 clúster 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. (Almaguer, y otros, 2008)

Hoy en día las llamadas supercomputadoras son equipos excesivamente caros mientras que la implementación de un clúster resulta más barata y económica. Se puede confeccionar un clúster 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 clúster 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.

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

1.1.2 Alto Rendimiento

Un clúster de alto rendimiento (*High Performance*) se utiliza para conseguir altas prestaciones en cuanto a capacidad de cálculo. El funcionamiento básico de este tipo de clúster consiste en reducir el tamaño de una tarea en un conjunto de tareas más pequeñas y repartirla entre todos sus nodos. Así logra resolver una tarea en el menor tiempo posible consiguiendo un alto rendimiento en el procesamiento de datos. (Montero, 2008)

1.1.3 Alta Disponibilidad

Un clúster de alta disponibilidad (*High Availability*), tiene el objetivo de proporcionar máxima disponibilidad y confiabilidad de los servicios que ofrece de tal manera que estos se brinden ininterrumpidamente. Este tipo de clúster se compone por al menos dos servidores. En su forma más básica un servidor, el primario, se encuentra prestando el servicio, mientras que el secundario se mantiene supervisándolo constantemente para si falla, tomar su lugar sin afectar el correcto funcionar de la aplicación. (Montero, 2008)

1.1.4 Balanceo de Carga

Un clúster de balanceo de carga (*Load Balancing*) se utiliza en entornos donde es necesario realizar muchas tareas pequeñas. Se compone por dos tipos de nodos, los balanceadores de carga y los servidores reales. Su principio de funcionamiento es repartir las tareas entre todos los servidores reales que componen el clúster. Este tipo de clúster es altamente escalable y su arquitectura facilita realizar implementaciones de alta disponibilidad. (Montero, 2008)

La solución final propuesta es una combinación de un clúster de alto rendimiento y de alta disponibilidad.

1.2 Técnicas de clustering para bases de datos en el mundo

A nivel mundial existe una gran variedad del empleo de clúster para servidores de bases de datos. Entre los ejemplos más sobresalientes se encuentra el clúster de *Google*, siendo actualmente el motor de búsqueda de Internet más grande y usado. 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 clúster y reduce las posibilidades de ocurrencia de fallos, por otra parte aumenta la escalabilidad y el rendimiento del sistema.

Arquitectura de un RAC (Real *Application* Clúster).

Una configuración *Oracle* 10g RAC mejora los tiempos de respuesta de su aplicación para satisfacer una necesidad de procesamiento creciente. A medida que usted agrega recursos, Real *Application* Clúster puede utilizarlos y extender su poder más allá de los límites de componentes individuales.

Además, la tecnología de *Oracle*, permite mejorar el tiempo de ejecución de sus consultas gracias al uso de consultas en paralelo en diferentes nodos (*multinode parallel query*). Un ambiente adecuadamente configurado tolera fallas con un tiempo de caída mínimo.

El servicio de TAF (*Transparent Application Failover*), de recuperación transparente ante fallas, permite que ante una caída en un servidor del clúster los usuarios de la aplicación no sean afectados. *Oracle* 10g RAC es un componente fundamental en soluciones de alta disponibilidad.

En Cuba.

En Cuba no está difundido el uso de la tecnología clúster para servidores de de bases de datos, aunque se emplean técnicas de clúster con otros fines, como por ejemplo lo es el clúster que emplea ETECSA y la Aduana bajo *software* propietario.

Otro ejemplo es el Centro de Ingeniería Genética y Biotecnología, que cuenta con un clúster conformado por 128 procesadores, sistema dedicado a numerosas tareas dentro de la investigación científica de la institución. (Almaguer, y otros, 2008)

1.3 PostgreSQL

PostgreSQL es la base de datos relacional de código abierto más avanzada del mundo. Distribuida bajo licencia BSD (del inglés, *Berkeley Software Distribution*), lleva más de 15 años desarrollándose y su arquitectura goza de una excelente reputación por su fiabilidad, integridad de datos y correctitud.

PostgreSQL dispone de versiones para prácticamente todos los sistemas operativos y cumple totalmente con ACID (del inglés, *Atomicity, Consistency, Isolation, Durability*). Tiene soporte para claves extranjeras, *joins*, vistas, disparadores y procedimientos almacenados (en múltiples lenguajes de programación).

Incluye la mayoría de los tipos de datos de SQL92 y SQL99 y, asimismo, soporta el almacenamiento de grandes objetos binarios, como imágenes, sonidos y vídeos. Tiene interfaces de programación nativas para C/C++, *Java*, *.Net*, *Perl*, PHP, *Python*, *Ruby*, Tcl y ODBC, entre otros, y una excepcional documentación.

PostgreSQL ofrece sofisticadas características tales como control concurrente multiversión (MVCC), *point in time recovery* (PITR), *tablespaces*, replicación asíncrona, transacciones anidadas (*savepoints*), copias de seguridad en caliente/en línea, un sofisticado Planificador/optimizador de consultas y *write ahead logging* para ser tolerante a fallos de *hardware*. Soporta juegos de caracteres internacionales, codificaciones de caracteres *multibyte*, *Unicode* y realiza ordenaciones dependiendo de la configuración de idioma local, de la diferenciación de mayúsculas y minúsculas y del formato.

Es altamente escalable tanto en la cantidad bruta de datos que puede manejar como en el número de usuarios concurrentes que puede atender. Hay sistemas activos en producción con PostgreSQL que manejan más de 4 *terabytes* de datos.

1.4 Herramientas utilizadas para balancear las cargas en un clúster de alta disponibilidad y alto rendimiento.

Los clúster que se implementan para obtener un alto rendimiento y una alta disponibilidad usan varias tecnologías para ganar un nivel extra de fiabilidad en un servicio, las máquinas deben estar conectadas entre sí por enlaces de red o series redundantes, para obtener tolerancia ante fallas. Cuando un servidor maestro se viene abajo, un servidor secundario toma control de los servicios.

A continuación alguna de las herramientas que se utilizan para balancear las peticiones del cliente hacia los distintos nodos de bases de datos por lo que se compone un clúster:

1.4.1 CyberCluster

Herramienta multimaster que presenta una replicación sincrónica basada en PostgreSQL 8.2. Proporciona balanceo de carga, asegurándose de que todos los nodos de la base de datos se pueden utilizar durante la operación normal. Esto es especialmente importante cuando se trata de la lectura de alto rendimiento.

CyberCluster en su arquitectura presenta un balanceador de carga, un servidor de base de datos, y un servidor de replicación. (Neustadt, 2009)

El servidor de replicación toma las peticiones de un nodo de la base de dato y replica los cambios a todos los nodos de la misma que componen el sistema, el servidor balanceador de carga es usado para distribuir la carga dentro del clúster Si no se dispone de balanceadores de carga, las aplicaciones pueden conectarse a cualquier nodo de la base de dato dentro del sistema.

CyberCluster contiene su propio equilibrador de carga que puede ser utilizado para distribuir la carga dentro del clúster. La carga en el sistema está determinada por el número de consultas activas. La máquina con el menor número de consultas activas es elegida para realizar una nueva solicitud.

Si el balanceador de carga detecta un problema en un nodo de la base de datos activa, automáticamente separa la base de datos del mismo (Neustadt, 2009)

Esta herramienta es lenta a la hora de distribuir las peticiones y requiere de una versión modificada del servidor. Otro problema que presenta es que cuando se replican objetos grandes deben ser colocados en un directorio que pueda ser leído por todos los nodos de la BD. (Reingart, 2009)

1.4.2 PgCluster

PgCluster es un sistema de replicación sincrónico multimaestro para PostgreSQL. Consiste en tres tipos de servidores, un servidor para balance de carga, un clúster de base de datos y un servidor de réplica. (Pupo, 2009)

Tiene dos funciones principales:

Compartir carga

- La carga de la sesión de las demandas es distribuida. Es efectivo en aplicaciones *Web* donde existe gran demanda por el número de peticiones.

Alta disponibilidad

- Cuando ocurre un fallo en el Clúster DB (Base de dato), el servidor de balance de carga y el de replicación separan el fallo del sistema, y continúa el servicio que usa el DB (Base de dato) restante.
- El Clúster DB (Base de dato) cuando es reparado puede restaurarse dinámicamente a un sistema, sin detener el servicio.
- Los datos son copiados automáticamente a la DB (Base de dato) restaurada o añadidos desde otra DB

PgCluster presenta algunas desventajas como son:

- La replicación es sincrónica, si la conexión falla pueden ocurrir errores.
- Requiere una versión "parcheada" de PostgreSQL.
- Instalación y configuración compleja.
- Puede requerir configuraciones avanzadas de *hardware*.

Esta solución no se recomienda para entornos con altos niveles de escritura, pues su sistema de replicación se vuelve ineficiente en estas condiciones.

1.4.3 *Pgpool-II*

Es un *software* de código abierto que ofrece características adicionales de alta disponibilidad. Proporciona la capacidad para la gestión y el *pooling* de conexiones, lo que permite la recuperación ante fallos en los servidores permitiendo configuraciones de alta disponibilidad. También incluye facilidades de replicación, permitiendo dos modos diferentes para su implementación. (Sabater, 2008)

En el modo Replicación *Pgpool-II* actúa como agente de replicación, para lo que envía las consultas de modificación de datos a todos los nodos del clúster y las de selección las distribuye entre ellos. Este método tiene el inconveniente de generar inconsistencias entre las bases de datos de los diferentes nodos cuando se utilizan funciones volátiles. (Sabater, 2008)

Pgpool-II es transparente tanto para el servidor como para el cliente, debido a que él crea una máscara en el servidor de aplicaciones de BD (*frontend*) por donde el cliente es capaz de conectarse y *Pgpool* distribuye las peticiones a los servidores de BD reales (*backend*).

Pgpool-II mantiene abiertas las conexiones a los servidores PostgreSQL y las reutiliza siempre que se solicita una nueva conexión con las mismas propiedades (nombre de usuario, base de datos y versión del protocolo). Ello reduce la sobrecarga en las conexiones y mejora la productividad global del sistema.

Si se replica una base de datos, la ejecución de una consulta *SELECT* en cualquiera de los servidores devolverá el mismo resultado. Aprovecha la característica de replicación para reducir la carga en cada uno de los servidores PostgreSQL distribuyendo las consultas *SELECT* entre los múltiples servidores, mejorando así la productividad global del sistema. En el mejor caso, el rendimiento mejora proporcionalmente al número de servidores PostgreSQL.

El balanceo de carga funciona mejor en la situación en la cual hay muchos usuarios ejecutando muchas consultas al mismo tiempo. Esto es válido para el método de replicación, siendo efectivo cuando son muchas pequeñas consultas.

Pgpool-II puede gestionar múltiples servidores PostgreSQL. El uso de la función de replicación permite crear una copia en dos o más discos físicos, de modo que el servicio puede continuar sin parar los servidores en caso de fallo en algún disco.

Al usar la función de paralelización de consultas, los datos están particionados y pueden dividirse entre varios servidores, de modo que la consulta puede ejecutarse en todos los servidores de manera concurrente para reducir el tiempo total de ejecución. La paralelización de consultas es una solución adecuada para búsquedas de datos a gran escala. (Sabater, 2008)

1.5 Herramientas para la alta disponibilidad en el clúster

En la actualidad las organizaciones dependen cada vez más de sus sistemas de información, y como es obvio se desea que estos sean seguros y permanezcan disponibles el mayor tiempo posible. (Paulo Clavijo, 2010)

La disponibilidad es una medida relativa a la preparación para su utilización de un sistema informático, mientras que la fiabilidad es una medida relativa a su capacidad para mantenerse operativo en el tiempo sin ningún tipo de fallo.

Los fallos potenciales de un sistema son los errores de componentes *hardware*, los errores o bloqueos del sistema operativo, los errores de las aplicaciones.

El proyecto *Linux-HA* (Alta Disponibilidad) tiene como objetivo proporcionar una solución de alta disponibilidad (*clustering*) para *Linux* que promueva la fiabilidad y disponibilidad de sistemas a través de su comunidad de desarrolladores.

Linux-HA (Alta Disponibilidad) se utiliza ampliamente y como una parte muy importante, en muchas soluciones de Alta Disponibilidad. Desde que comenzó en el año 1999 a la actualidad, sigue siendo una de las mejores soluciones de *software* HA (Alta Disponibilidad) para muchas plataformas. (Paulo Clavijo, 2010)

1.5.1 Heartbeat

La herramienta *heartbeat* es uno de los componentes principales del proyecto *Linux High Availability*. El objetivo fundamental del proyecto *Linux*-HA (Alta Disponibilidad) es desarrollar una solución de alta disponibilidad (*clustering*) para *Linux* que proporcione y promocione la fiabilidad, la disponibilidad y la calidad de servicio.

Permite implementar clústeres de control descentralizado, es estable, flexible y eficiente, requiere de dos máquinas como mínimo para su implementación, trabaja enviando latidos (ping) verificando que el servidor principal esté activo, estos envíos constantes de ping requieren de una respuesta por parte del servidor, cuando transcurre un tiempo y el servidor principal no responde *heartbeat* determina que el mismo está fuera de servicio o está inactivo, automáticamente éste activa el servidor secundario de forma que asuma todas las peticiones del cliente sin afectar la aplicación. (Suarez, 2002)

1.5.2 OpenAIS

OpenAIS Clúster *Framework* es una implementación *open source* de la *Application Interface Specification* (AIS). Un conjunto de especificaciones para estandarizar el desarrollo de servicios e interfaces para la alta disponibilidad, desarrolladas por el *Service Availability Forum* (Clavijo, 2010)

Los principales beneficios de una solución de Clúster HA (Alta Disponibilidad) basado en las normas AIS (*Application Interface Specification*) son la mejora en portabilidad e integración, permite sistemas más escalables, la reducción de costes y reutilización de componentes.

Esta estandarización puede ser muy beneficiosa no sólo para los componentes principales del *software* o *middleware* de *clustering*, si no por el hecho de que el clúster sea capaz de monitorizar un mayor número de servicios y recursos con un API (*Application Programming Interface*) unificada.

El proyecto *OpenAIS* implementa actualmente los componentes de infraestructura y membrecía. Y es utilizado en soluciones completas de clustering como *Pacemaker* o *Red Hat* Clúster. (Clavijo, 2010)

1.5.3 Proyecto Red Hat Cluster

RedHat-Clúster es un proyecto de desarrollo *open source* de diferentes componentes de clustering para *Linux*. Esta promovido principalmente por *RedHat*, ya que en dicho proyecto se basa casi en totalidad su producto *Red Hat* Clúster *Suite* para su distribución comercial *Linux* RHEL (*Red Hat Enterprise Linux*).

RedHat-Clúster es un conjunto de componentes que forman una solución de *clustering* HA (Alta Disponibilidad) completa. Tiene varias versiones, conocidas como: Clúster1, Clúster2 (versión estable y utilizada en *RHLE5*), y la tercera generación Clúster3. Todas ellas tienen en común que utilizan un CRM (*Customer Relationship Management*) propio llamado CMAN. (Clavijo, 2010)

La arquitectura de Clúster2 (segunda generación de *RedHat*-Cluster), se basa en líneas generales en el uso de *OpenAIS* como componente de mensaje/membrecía y CMAN como administrador de recursos (CRM). Así como otros componentes que proporcionan *fencing*, balanceo de carga, o las propias herramientas de administración del clúster.

1.6 Pruebas necesarias para el clúster.

Anteriormente se explicó las características de cada tipo de clúster y el objetivo para el cual fueron concebidos, alguno de estos se implementan para lograr alto rendimiento, mientras que otros para obtener alta disponibilidad, en este caso particular se hará una mezcla de ambos, para obtener una mayor capacidad de procesamiento y alta disponibilidad.

A continuación se detallarán un grupo de pruebas necesarias para la validación de la propuesta de solución genérica.

Pruebas de rendimiento

Son pruebas que se realizan para determinar la rapidez con que se realiza una tarea en el sistema en condiciones particulares de trabajo. Sirve además para validar y verificar otros atributos de la calidad del sistema, tales como uso de recursos, escalabilidad. En general

es una prueba que se esfuerza por medir que partes del sistema o de carga de trabajo provocan que haya mal rendimiento.

1.6.1 Pruebas de carga

Este tipo de prueba es una subcategoría de las pruebas de rendimiento, que se ejecuta para comprender el comportamiento de una aplicación ante una carga determinada. Esta carga puede estar determinada por la conexión de un número de usuarios conectados de manera concurrente al sistema en un tiempo determinado.

El resultado de esta prueba nos dará el tiempo de respuesta de todas las transacciones críticas. Monitorizando la base de datos y el servidor de aplicación, se puede mostrar problemas en la aplicación. (Testhouse, 2009)

Pruebas de configuración

Estas pruebas se realizan para garantizar que la aplicación funcione correctamente sobre diferentes configuraciones de *hardware* y/o software. Durante su realización tiende a ocurrir un ciclo de perfeccionamiento en el diseño de la solución.

1.6.2 Pruebas de benchmark

Este tipo de prueba tiene 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.

Para realizar las pruebas *benchmark* y carga de es necesario utilizar herramientas que permitan generar carga. (Palmares, y otros, 2009)

1.7 Herramientas para pruebas de benchmarking

1.7.1 Tsung

Herramienta que simula un elevado número de usuarios, así como, sesiones diferentes en una base de datos, está desarrollado en *Erlang* que es un lenguaje diseñado para

desarrollar sistemas altamente concurrentes y tolerantes a fallos, robusta y fiable, herramienta de código abierto. (Prada, 2006)

1.7.2 Pgbench

Prueba práctica que se incluye dentro de PostgreSQL, prueba de rendimiento muy simple con la que se puede comprobar el subsistema y/o la velocidad a la que se procesan las conexiones. Útil para demostrar problemas importantes de *hardware* ó en el sistema operativo. Por defecto, las pruebas *pgbench* son un escenario que se inspira en TPC-B, que implica a cinco *SELECT, UPDATE*, y comandos *INSERT* por transacción. (Documentación de PostgreSQL 9.0devel, 2009)

1.7.3 Jmeter

Se destaca por su versatilidad, estabilidad, y por ser de uso gratuito, desarrollada por la *Apache Software Foundation*, se trata de una aplicación 100% *Java* que se utiliza habitualmente para realizar pruebas funcionales y medir rendimientos. Fue originalmente diseñada para testear aplicaciones *web*, pero desde entonces ha evolucionado incrementando su funcionalidad y los tipos de pruebas a los que puede aplicarse.

Puede utilizarse para simular condiciones de carga muy elevada en servidores, redes o aplicaciones concretas, para comprobar su capacidad o analizar el rendimiento general bajo diferentes condiciones de carga. (SoftQaNetwork, 2010)

Muestra los resultados de las pruebas en una amplia variedad de informes y gráficas. Además facilita a una rápida detección de los cuellos de botella existentes debido al tiempo de respuesta excesivo.

La herramienta seleccionada para realizar las pruebas de rendimiento ha sido *Apache Jmeter*, debido a todas las facilidades que la misma proporciona.

1.8 Métricas que se utilizan

Para evaluar el rendimiento de una aplicación, es necesario hacer mediciones del comportamiento de algunas de sus características. Cuando se realizan mediciones, se puede recoger información referente al funcionamiento de todo el sistema y/o de los servidores que lo conforman. Las más comunes son:

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

Cuando se trata de analizar el comportamiento de los servidores que forman parte del sistema las mediciones se enfocan en el consumo de los recursos. Las mediciones que normalmente se realizan en los servidores son:

- % de uso del CPU.
- % de consumo de memoria RAM.
- % de uso de la red.

1.9 Requerimientos mínimos para la solución

El clúster está formado por un balanceador de carga y dos servidores reales conectados entre sí por una red local. Los servidores son computadoras personales equipadas con un procesador *Pentiun* 4 a 3.00 GHz, 1 GB de memoria *RAM*, disco duro *SATA* de 160 *GB* y una tarjeta de red de 1.0 Gbps.

Se instalará la distribución *Debian/Linux* 5.0 del sistema operativo *Linux* en todos los nodos del clúster. Esta instalación no incluye el entorno gráfico y se configurará para utilizar el idioma inglés y la zona horaria de Cuba. También debe tener configurada las fuentes APT (*Advanced Packaging Tool*).

1.9.1 Requerimientos de Hardware.

- 1 computadora para el balanceador de carga primario.
- 1 computadora para el balanceador de carga de respaldo.
- 2 computadoras para los servidores de bases de datos.
- 1 computadora para el cliente.

1.9.2 Requerimientos de Software

- *Pgpool*-II-2.2.5.
- *Heartbeat*-2.
- PostgreSQL-8.4.

Otros requerimientos

Red de área local con velocidad igual o superior a 100 Mbps

Jmeter

Conclusiones del capítulo

Durante el desarrollo de este capítulo se realizó un estudio de varias soluciones con *software* libre existentes en el mundo para implementar un clster de servidores de bases de datos.

Después de haber realizado un estudio de las técnicas de clúster de alto rendimiento y alta disponibilidad respectivamente, además del conjunto de herramientas libres que se utilizan para su implementación, se definió un conjunto de pruebas para validar la propuesta de solución basada en este tipo de clúster. Se concluye que:

La implementación del clúster de alto rendimiento y alta disponibilidad que constituye la base de la propuesta de solución, se realizará utilizando la combinación de la herramientas libres *Pgpool-II* como balanceador de carga y *Heartbeat* para implementar la alta disponibilidad entre los balanceadores de carga. Es necesario realizar pruebas de configuración y carga para validar dicha propuesta de solución.

Capítulo 2. Propuesta de diseño de la solución de clúster.

Este capítulo está centrado en el diseño de una propuesta de solución genérica en la cual se confeccionará un diseño lógico y físico, además de todo el tema referente a copias de seguridad, recuperación ante fallos y la seguridad.

2.1 Diseño de la solución.

Se confeccionó una propuesta de solución para la puesta en marcha de un clúster de servidores de bases de datos. Los nodos utilizados para conformar el clúster pueden ser computadoras personales con características de *hardware* variables. La ubicación física del mismo debe realizarse en una instalación con buena climatización y acceso restringido al personal.

A continuación se describe la arquitectura de la solución propuesta.

2.1.1 Diseño lógico.

Para la implementación de un clúster de alta disponibilidad y alto rendimiento, se propone que el mismo sea implementado sobre el sistema operativo *GNU/Linux* distribución *Debian Lenny*, esta instalación no incluirá el entorno gráfico y se configurará para utilizar idioma inglés y la zona horaria de cuba, teniendo configurada además las fuentes APT (*Advanced Packaging Tool*). Se utiliza como sistema de gestión de bases de datos a PostgreSQL-8.4, además de utilizar a *Pgpool-II*-2.2.5 para el balanceo de carga y *Heartbeat*-2 como herramienta para implementar la alta disponibilidad.

El clúster está compuesto por la siguiente arquitectura (ver Fig. 1)

- Un balanceador de carga primario.
- Un balanceador de carga secundario de respaldo.
- Dos servidores de bases de datos PostgreSQL, como mínimo dos.

Todas las sentencias SQL que envía el cliente al servidor, pasan por el nodo *Pgpool-II* el cual tiene como acción fundamental, repartir las peticiones entrantes entre cada uno de los nodos-SQL disminuyendo así la carga entre los mismos.

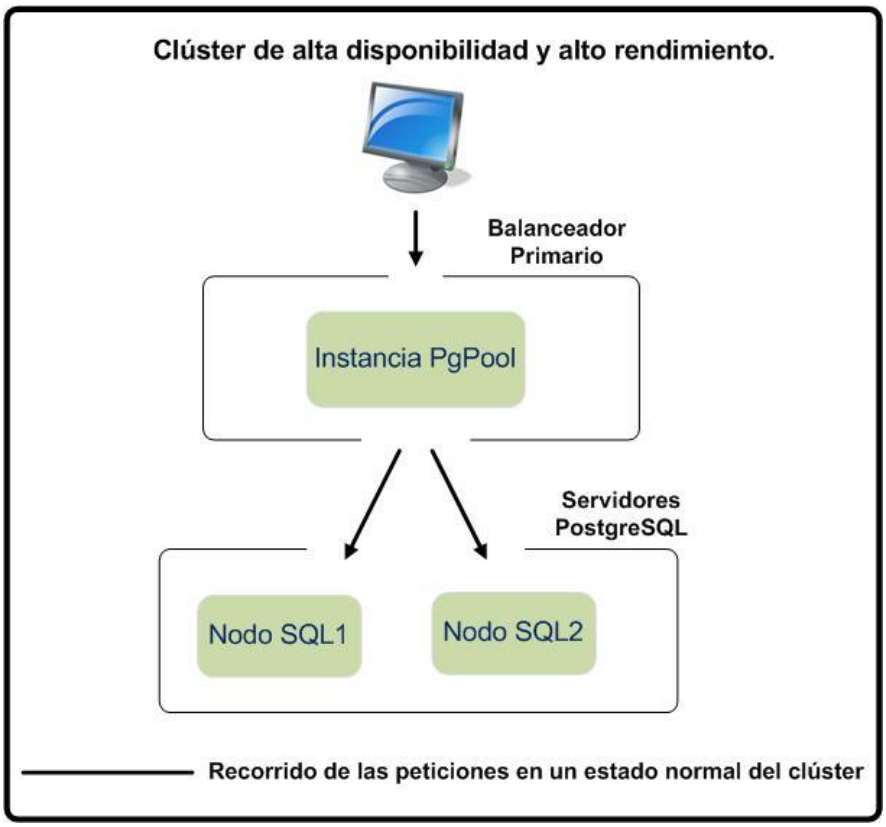


Figura 1. Arquitectura del clúster.

Las consultas de lectura que lleguen a dicho balanceador serán enviadas hacia cualquier servidor ya que todos disponen de la misma información, mientras que toda acción de modificación de datos se ejecutará automáticamente en cada uno de los servidores, manteniendo de esta manera todos los servidores sincronizados con la misma información.

Esta propuesta de diseño de solución, contiene un posible punto de fallas, debido que se puede desplomar un nodo-SQL, no imposibilitando esta acción el correcto funcionar de la misma, puesto que las peticiones se reenvían hacia el otro nodo-SQL (ver Fig. 2)

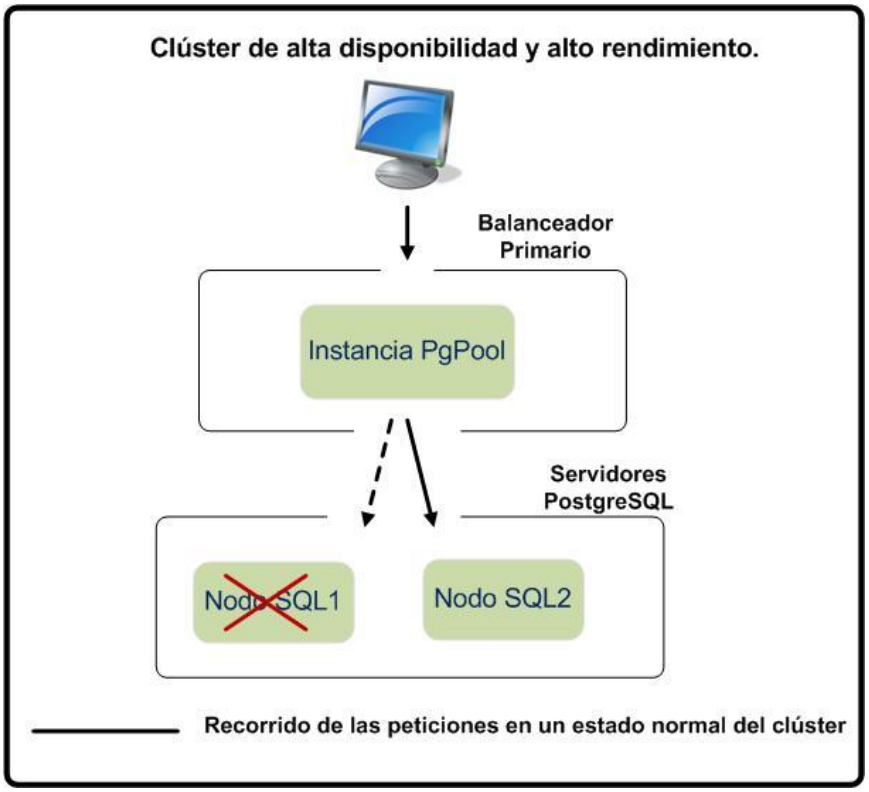


Figura 2. Representación lógica cuando falla un nodo SQL.

Con la arquitectura actual que presenta el clúster si el balanceador de carga queda fuera de servicio, el clúster completo también lo hace, convirtiendo esto al balanceador de carga en un punto único de fallas (ver Fig. 3)

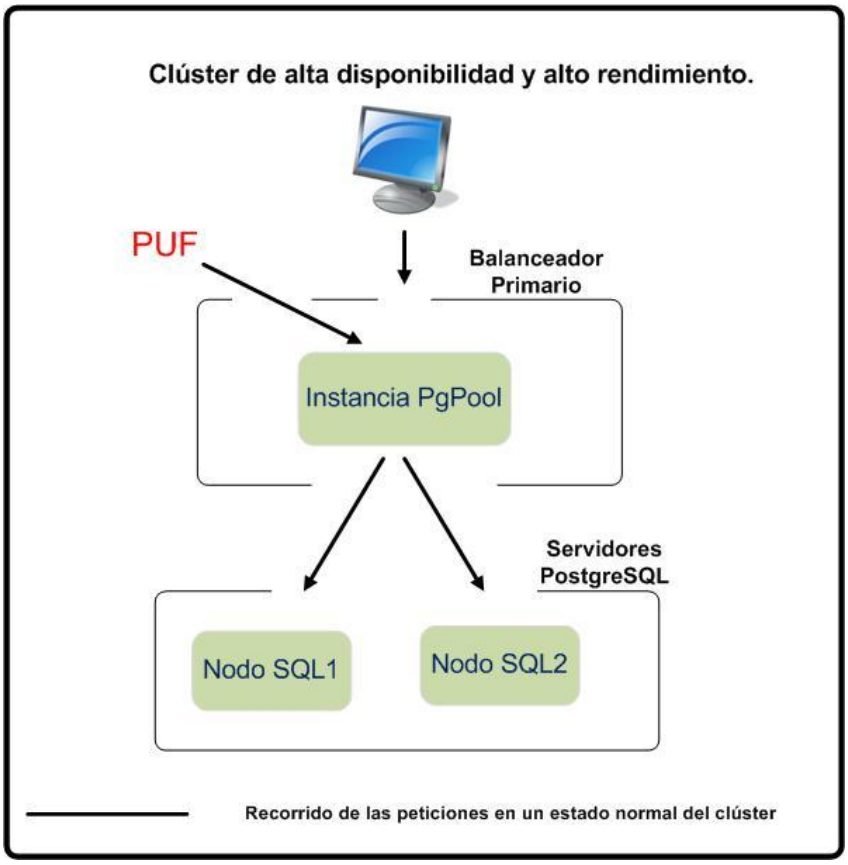


Figura 3. Representación lógica del nodo primario.

Para evitar esto se configura un balanceador de carga de respaldo, implementando así la alta disponibilidad que contiene *Pgpool-II* dentro de su configuración, en este caso el balanceador que actúa como nodo de respaldo, toma las actividades del nodo primario en una posible caída del mismo, siendo este proceso completamente transparente para el usuario. El balanceador de carga de respaldo se montará en un nuevo servidor que será agregado al clúster (ver Fig. 4)

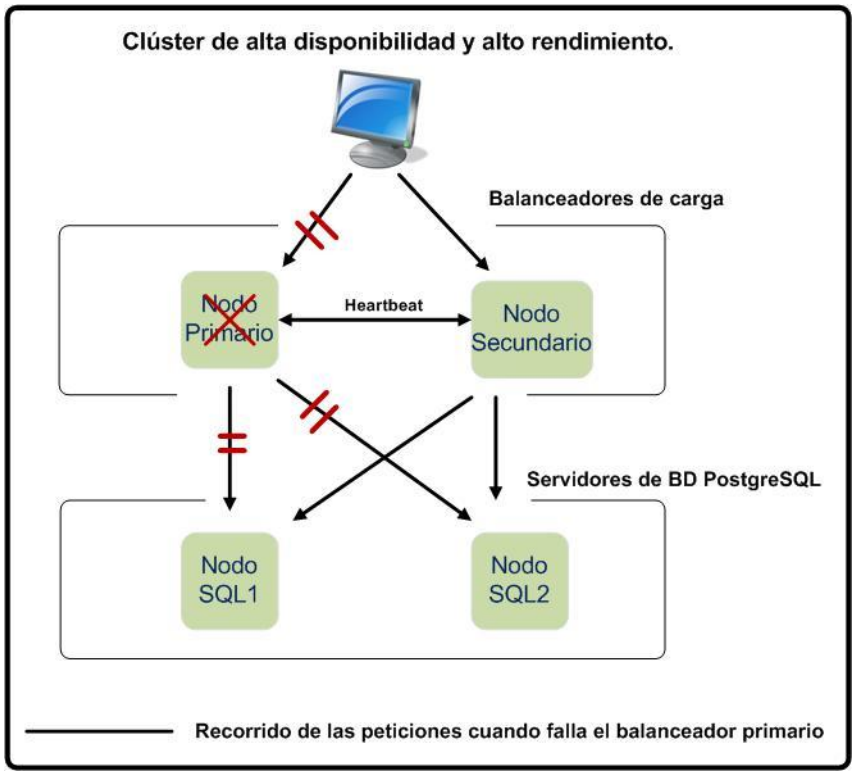


Figura 4. Representación lógica cuando falla el balanceador primario.

Luego de restablecido el servidor principal, toma nuevamente el control, quedando el de respaldo una vez más en espera de una recaída, volviendo a su normalidad el clúster (ver Fig. 5)

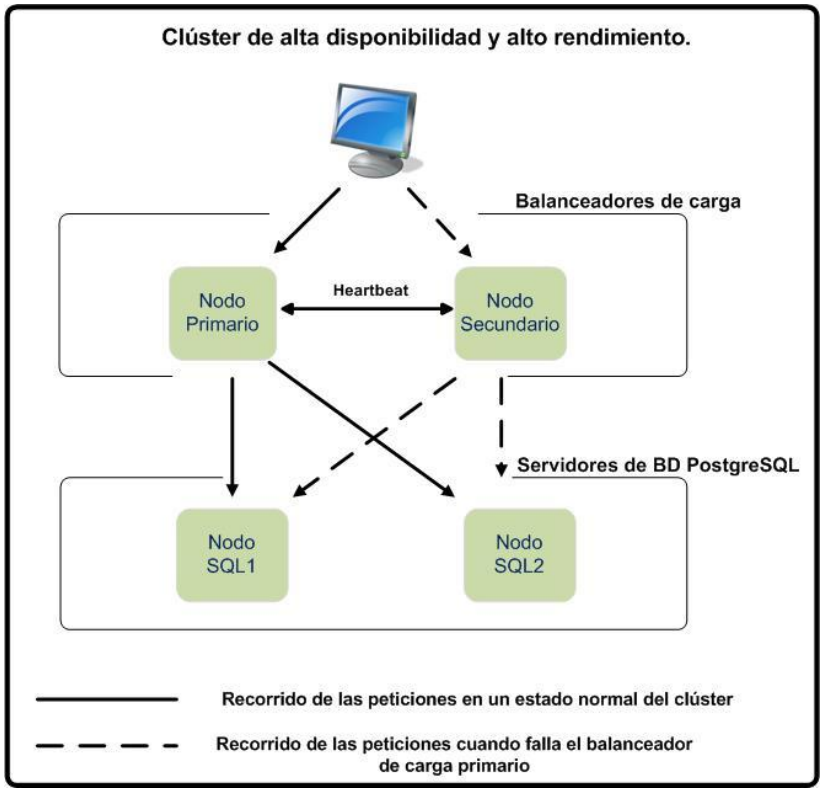


Figura 5. Representación lógica general del clúster.

Para la implementación de la solución son necesarias las siguientes herramientas:

- *Pgpool-II* (versión utilizado 2.2.5)
- *Heartbeat* (versión utilizada 2)
- Servidor de base de datos PostgreSQL (versión utilizada 8.4)

2.1.2 Diseño físico.

La arquitectura propuesta para el clúster está conformada por dos nodos, encargados de fluctuar las peticiones entre los servidores reales, instalándose en cada uno de ellos la herramienta *Pgpool*-II-2.2.5, encargada de proporcionar el balanceo de carga además de implementar un mecanismo de recuperación ante fallas, ante el desplome de alguno de los servidores.

El sistema de gestión de bases de datos empleado es PostgreSQL-8.4, ya que el mismo dispone de versiones para prácticamente todos los sistemas operativos, soporta el

almacenamiento de grandes objetos binarios, como imágenes, sonidos y vídeos, ofrece sofisticadas características tales como control concurrente multiversión, *point in time recovery* (PITR), replicación asíncrona, copias de seguridad en caliente/en línea, un sofisticado planificador/optimizador de consultas y *write ahead logging* para ser tolerante a fallos de *hardware* .

El diseño físico más sencillo es el caso en que el balanceador de carga principal está trabajando en estado normal (ver Fig. 6)

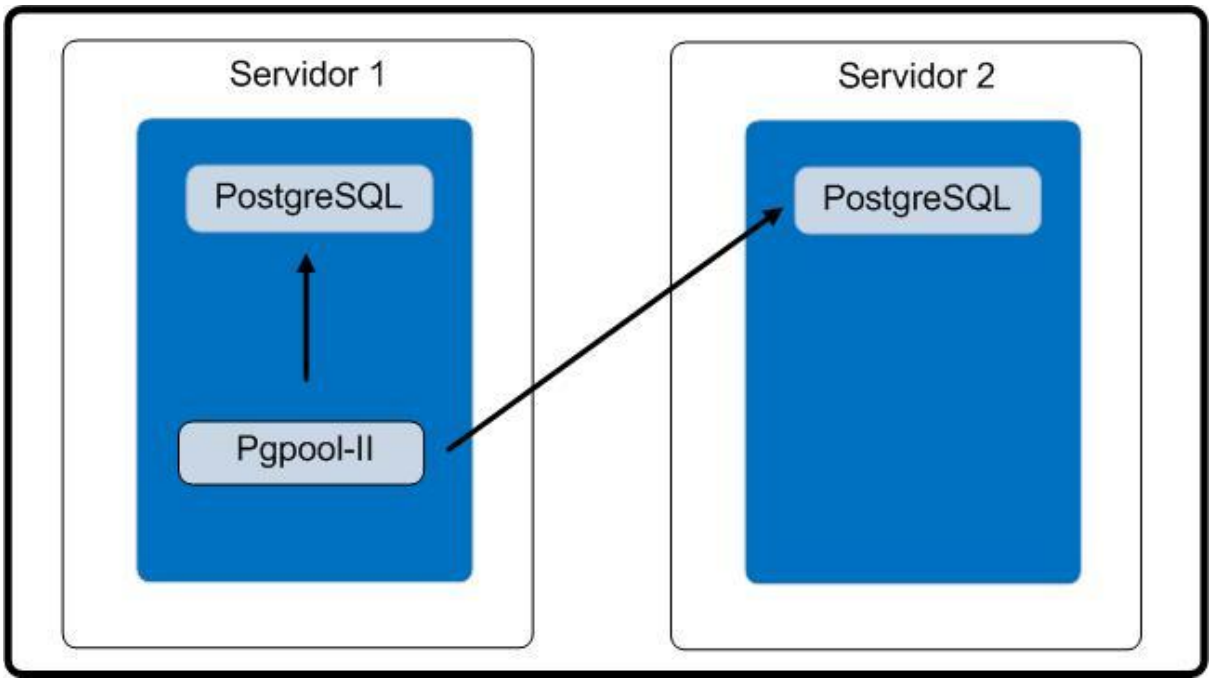


Figura 6. Representación física del balanceador de carga primario.

La afectación que presenta este diseño es que la instancia del nodo SQL caiga, pues esto no afectaría en nada la solución puesto que las peticiones se direccionan hacia el otro nodo SQL (ver Fig. 7)

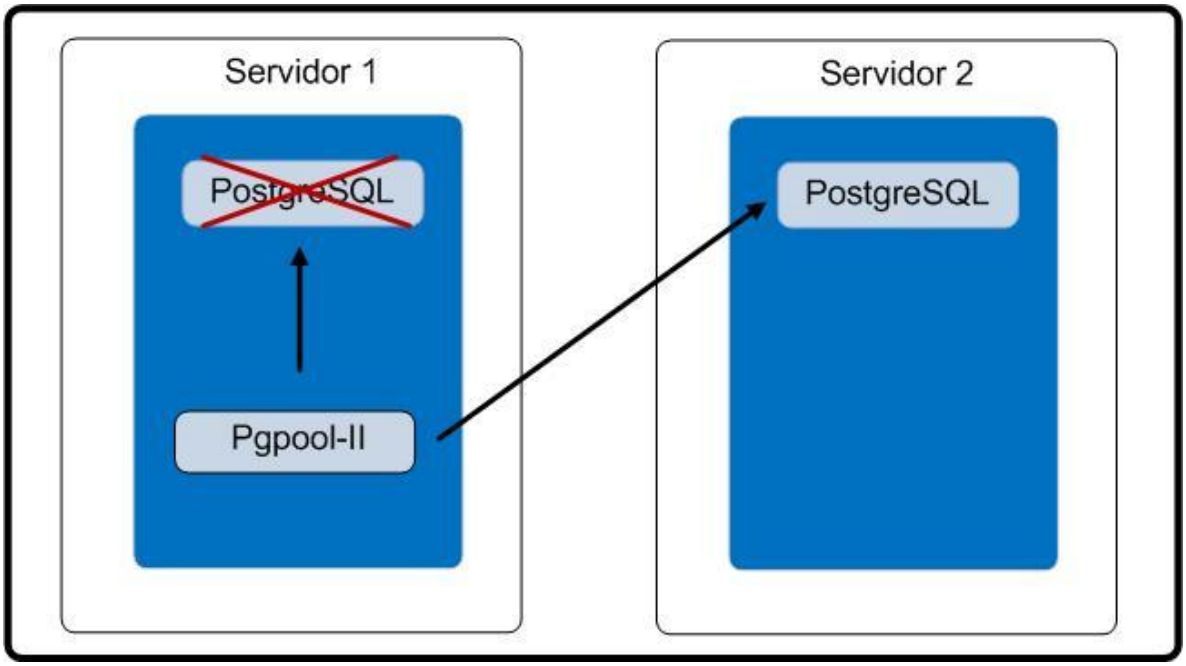


Figura 7. Representación física cuando falla un nodo SQL.

Este diseño presenta como consecuencia un punto de fallas, ya que el nodo *Pgpool-II* puede salir de servicio e imposibilitar la entrada de sentencias SQL por parte del cliente hacia el servidor. Ante esta falla y gracias a la implementación de alta disponibilidad con *Pgpool-II*, el servidor secundario que contiene al nodo de respaldo, toma las acciones del servidor principal y continúa prestando servicios sin afectar al cliente (ver Fig. 8)

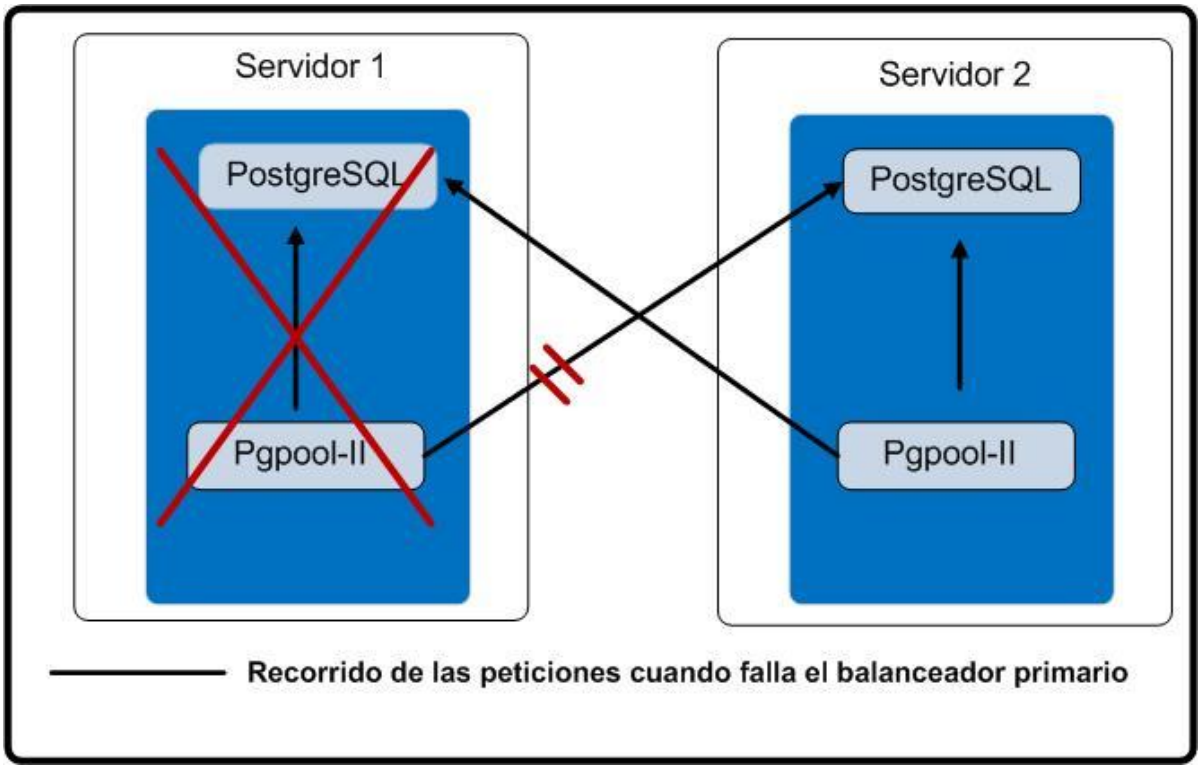


Figura 8. Representación física cuando falla el balanceador primario.

Pgpool-II brinda la posibilidad de seguir dando servicio tras el fallo de N-1 servidores de PostgreSQL en el clúster, para ello utilizaremos *Heartbeat*, dicha herramienta permite detectar la caída completa de un nodo, por lo que entra en la conformación del clúster un nueva dirección IP que sería la dirección IP de servicio, la cual ambos nodos van a compartir y la cual *Heartbeat* se encargará de gestionar.

La dirección IP de servicio es gestionada por el sistema de *Heartbeat*, es movida por el clúster donde los servicios correspondientes se estén ejecutando. Estas direcciones de servicio son direcciones a través de las cuales los clientes y usuarios de los servicios en *Heartbeat* acceden a dichos servicios.

Es de suma importancia que la dirección IP de servicio no sea gestionada por el sistema operativo, solo que sea gestionada por *Heartbeat*. Si se le da una dirección administrativa a *Heartbeat* para que la gestione, esto causará problemas pues confundirá al sistema.

Acarreando como consecuencia que el sistema operativo y *Heartbeat* se peleen por el control de la misma.

Heartbeat, es un *software* que se utilizará para dar alta disponibilidad a *Pgpool-II*, soporta un modelo de dependencias muy sofisticado para clústeres de N nodos, es muy útil y estable (ver Fig. 9)

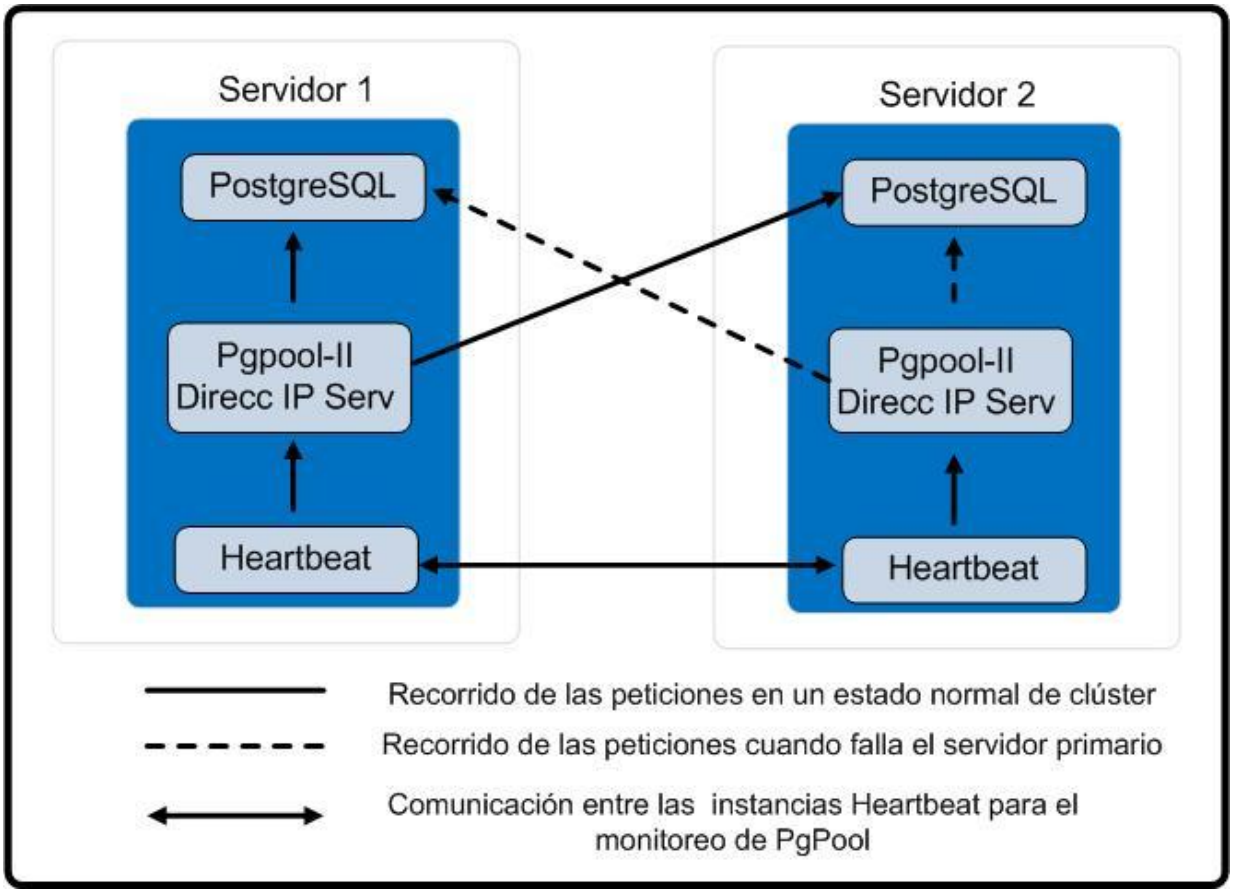


Figura 9. Representación física general del clúster.

2.1.3 Diseño de interconexión.

El diseño de interconexión que se propone para este tipo soluciones está compuesto por dos servidores, los cuales van a estar recepcionando todas las sentencias SQL que hace llegar el cliente al servidor, reenviándolas hacia los nodos SQL, en este caso el nodo balanceador de carga que recibe dichas peticiones es aquel que esta activado.

Estos nodos actúan como capa intermediaria entre el cliente y las bases de datos a las cuales estos nodos balanceadores enviarán las sentencias de modificación y solo enviarán

las de selección al nodo que no haya recibido alguna de estas solicitudes de selección, luego de ambos haber recibido dichas sentencias de selección pues se repite el proceso continuamente.

Cuando el nodo que está actuando de primario presenta alguna falla y deje de prestar servicio, automáticamente el nodo secundario toma las acciones de este, a través de la configuración de alta disponibilidad que ofrece *Pgpool-II*, siendo todo este proceso completamente transparente para el usuario.

Para poner en práctica esta configuración de alta disponibilidad, se utiliza el *software Heartbeat*, desarrollado por el proyecto *Linux-HA* (Alta Disponibilidad), éste puede llevar a cabo la detección de la caída de nodos., esta es una aplicación altamente escalable y de múltiples funciones (ver Fig. 10)

El diseño se interconexión entre ambos nodos esta propuesto para que se configuren tres tarjetas de red las cuales se describes a continuación:

- Una tarjeta de red: para que *Pgpool-II* se conecte con PostgreSQL
- Una tarjeta de red: para los Servidores
- Una tarjeta de red: para que *Heartbeat* monitoree los nodos balanceadores de carga

Una relación de confianza entre el balanceador de carga primario y los servidores reales es necesaria para que este pueda mandar a reiniciarlos cuando los recursos sean migrados a él, para esto es necesario que en cada servidor tener instalado el servicio ssh.

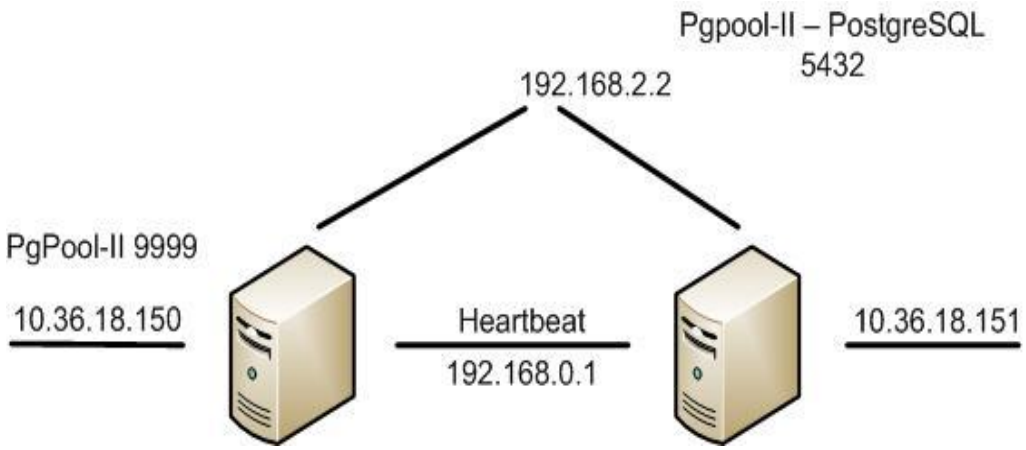


Figura 9. Representación del diseño de interconexión.

2.1.4 Diseño de seguridad.

En la actualidad uno de los puntos más críticos es el de la seguridad, muchos sistemas están expuestos, sin saberlo, a ataques, por lo que la mayoría de estos no disponen de una infraestructura de protección.

Con un *router* de *hardware* no es suficiente, puesto que dichos *router* rara vez son actualizados, por lo que se pierde su efectividad con el tiempo, incluso llegando a facilitar los ataques en vez de impedirlos.

Como primera medida de seguridad se propone la instalación de un cortafuego, que no es más que un programa que sirve para filtrar las comunicaciones de un ordenador o de una red, tanto entrantes como salientes, permitiendo o denegando estas comunicaciones en función de una serie de criterios.

La configuración correcta de un cortafuego se basa en conocimientos considerables de los protocolos de red y de la seguridad de la computadora. Errores pequeños pueden dejar a un cortafuego sin valor como herramienta de seguridad.

Un cortafuego puede ser de dos tipos diferentes:

Cortafuego de *hardware*: se trata de un tipo de cortafuego instalado en un periférico de aspecto parecido a un *router*. Es una muy buena solución cuando se habla de una red, ya

que permite hacer toda la configuración de cortafuego en un solo punto al que se conectan los ordenadores.

Cortafuego de *software*: es el más común y utilizado, se trata de un *software* que se instala en el ordenador, por lo que sólo va a proteger al ordenador en el que está instalado. (Josito, 2007)

La solución puede contar con un cortafuego de *hardware*, por lo que esto puede incurrir en costos adicionales para el sistema (ver Fig. 11)

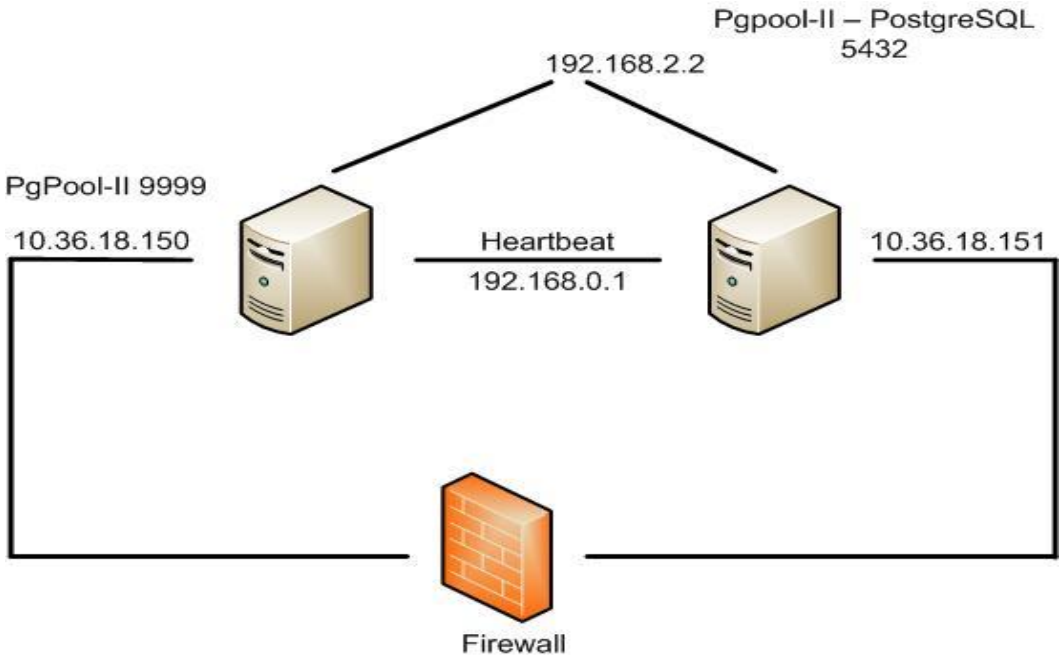


Figura 11. Diseño de seguridad con el cortafuego de *hardware*.

Para que el costo sea completamente aceptado por el cliente, se implementó de la siguiente el cortafuegos (ver Fig. 12), estando estos dentro del propio servidor.

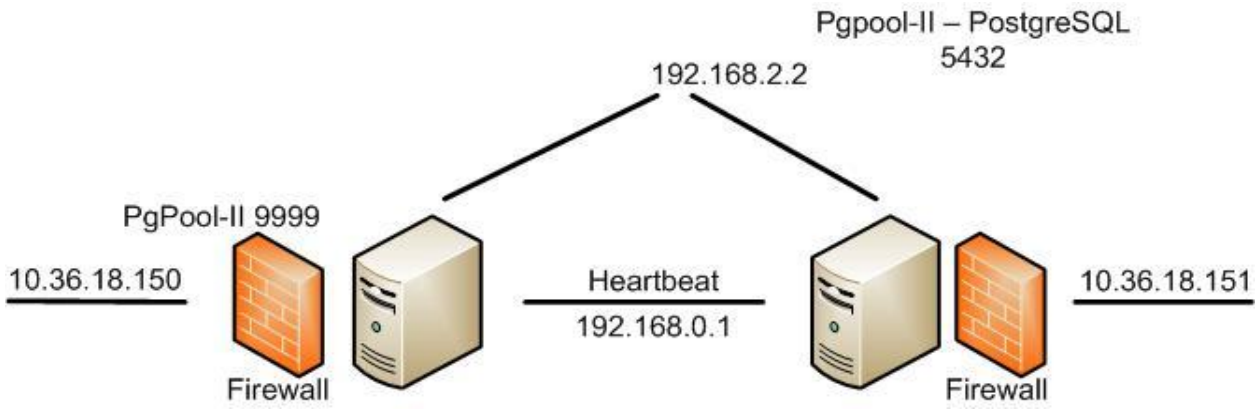


Figura 12. Diseño de seguridad con el cortafuego de *software*.

Conclusiones del capítulo

El diseño propuesto será desplegado en un escenario de pruebas en el laboratorio aplicándosele un grupo de pruebas para su validación.

El diseño incluyó además algunas consideraciones para el despliegue del clúster en su entorno real así como requerimientos de seguridad.

Capítulo 3. Diseño de las pruebas y análisis de los resultados.

Este capítulo está centrado en la descripción y análisis de resultados de las pruebas de carga realizadas a la solución. Tiene como objetivo validar la propuesta de solución.

3.1 Descripción de las pruebas

Con el objetivo de garantizar el correcto funcionamiento de cualquier sistema es necesario realizar un proceso de pruebas donde se demuestre que dicho sistema está listo para su despliegue. Este proceso de pruebas, se realizó con el propósito de demostrar que la aplicación funciona correctamente, permitiendo su utilización, una disminución en los tiempos de respuestas del sistema.

Estas pruebas se realizaron 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.)
- Establecer un grado de calidad del sistema.
- Proveer un entendimiento general de cómo funciona el mismo.
- Probar además que usable y operable.

Durante la realización de las pruebas se estudió el comportamiento de las funcionalidades de la solución, para ver si se desempeña como se previó en el diseño de la variante de la propuesta de solución. Las funcionalidades de la solución que se probaron fueron las que

definen el comportamiento del clúster y pueden afectar el correcto funcionamiento de la aplicación.

Para el caso particular de esta solución con el objetivo de comprobar el correcto funcionamiento de los sistemas de clúster de bases de datos y medir su rendimiento se decidió realizar un conjunto de pruebas las cuáles se clasifican fundamentalmente en dos tipos.

3.1.1 Pruebas de carga

Las pruebas de carga fueron utilizadas para validar y evaluar la aceptabilidad de los límites operacionales de un sistema bajo distintos volúmenes de trabajo mientras la solución de clúster se mantiene constante (la misma cantidad de nodos). Mientras que en la otra variante se le aplicó la misma prueba a un solo servidor PostgreSQL.

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 clúster 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* (Unidad Central de Procesamiento), por ciento de uso de memoria *RAM* (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 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 cómo se mejoran los tiempos de respuesta promedio de la aplicación desplegada sobre el clúster 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 cómo se deterioran los tiempos de respuesta promedio a medida que se vayan aumentando los niveles de concurrencia.

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

Muestra además los resultados de las pruebas en una amplia variedad de informes y gráficas, facilita a una rápida detección de los cuellos de botella existentes debido al tiempo de respuesta excesivo.

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 clúster.

3.1.2 Pruebas de benchmark

Estas pruebas realizan para comparar el comportamiento de los sistemas o las aplicaciones sobre diferentes configuraciones de *software* y *hardware*. 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.

3.2 Pruebas de configuración

Descripción del despliegue:

Clúster de alto rendimiento y alta disponibilidad para bases de datos, compuesto por dos balanceadores de carga y dos servidores de bases de datos.

El clúster incluye las siguientes funcionalidades:

- Envía las consultas de lectura (*SELECT)*, a todos los nodos SQL, balanceando las mismas entre aquellos que presenten mayor disponibilidad.
- Envía todas las consultas de modificación (*insert, update, delete*), hacia todos los nodos SQL
- Recuperación ante fallos en los servidores de bases de datos
- Alta disponibilidad para el servicio de balanceo de carga.

3.3 Funcionalidades del sistema a probar

- Balanceo de carga.
- Replicación.
- Alta disponibilidad en el balanceador de carga.
- Recuperación ante fallos de los servidores de bases de datos.
- Correcto funcionamiento de la aplicación.

3.4 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, la misma se realice 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 clúster sea transparente al usuario.

3.6 Pruebas al diseño genérico de la propuesta de solución

Para validar que este diseño se desempeña correctamente se realizaron pruebas de configuración a las principales funcionalidades del clúster. El listado de las pruebas realizadas y el propósito que cada una de estas se describe en la tabla 1.1.

En la tabla 1.1 Pruebas realizadas al diseño genérico del cluster de alto rendimiento y alta disponibilidad:

Referencia	Tipo	Propósito
DG-01	Configuración	Comprobar que el diseño genérico de la propuesta de solución realiza el balanceo de carga correctamente.
DG-02	Configuración	Comprobar que el diseño genérico de la propuesta de solución realiza la replicación de datos correctamente.
DG-03	Configuración	Comprobar que el diseño genérico de la propuesta de solución es capaz de recuperase ante fallos del balanceador de carga primario.
DG-04	Carga	Mostrar como disminuye el tiempo de respuesta del sistema cuando se aumenta el número de servidores de bases de datos en el clúster de alto rendimiento y alta disponibilidad.

Siguiendo las indicaciones del documento “Guía para la instalación y configuración de un cluster de alto rendimiento y alta disponibilidad genérico para servidores de bases datos”, se realizó el despliegue de un cluster genérico que estaba compuesto por un balanceador de carga primario, uno de respaldo y dos servidores de bases de datos.

Prueba de configuración DG-01

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 PostgreSQL

Se utilizó la herramienta *Jmeter* para grabar la conexión de dichos usuarios, la misma se repitió varias veces, simulando en cada vez 200 usuarios concurrentes. Como resultado se obtuvo a través del script “balanceo de carga”, la cantidad de usuarios que fueron balanceados a los distintos nodos de bases de datos por los que está compuesto el clúster.

Prueba de configuración DG-02

Para analizar cómo 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, 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 PostgreSQL.

Prueba de configuración DG-03

Para simular la caída del balanceador de carga primario, se desconectó la red, del servidor donde se encuentra el mismo. Observando el comportamiento del balanceador de carga de respaldo se pudo ver como detectó que el balanceador de carga primario estaba fuera de servicio.

Como respuesta a esto se configuró la herramienta *Heartbeat*, la cual a través de una dirección IP (*Internet Protocol*) de servicio monitorea constantemente los nodos balanceadores de carga, al ver que el balanceador de carga primario falló, pues esta a su vez le pasa todos los recursos al balanceador de carga de respaldo, siendo todo este proceso transparente para el usuario.

Prueba de carga DG-04

Para desarrollar las pruebas de carga en la solución, se montó un servidor PostgreSQL, en el cual mediante la herramienta *Jmeter*, se le simuló 200 usuarios conectados de manera concurrente para así observar su rendimiento.

Posteriormente se le aplicó la misma prueba al clúster, conformando por dos servidores PostgreSQL, disminuyendo el rendimiento en el mismo, ya que para el servidor solo de PostgreSQL se comete un error de un 10 % al recibir dicha cantidad de usuarios conectados concurrentemente.

Número de servidores de bases de datos fuera y dentro del cluster (c)	Rendimiento	Error	Atención de las peticiones en **ms**
1	50 peticiones/sec	10 %	1500
2 (c)	85 peticiones/sec	0 %	16

Análisis de los resultados

Durante las pruebas de configuración realizadas al diseño genérico del clúster de alto rendimiento y alta disponibilidad se probaron las funcionalidades: balanceo de carga y recuperación ante fallos de los balanceadores de carga.

Las pruebas de carga a las que se sometió la solución, presentó una mejora, puesto que el rendimiento mejora con el aumento de nodos de bases de datos en el clúster, la misma arrojó que para un solo servidor PostgreSQL este presenta un error del 10% al recibir la simulación de usuarios conectados concurrentemente, mientras que el clúster no presenta errores por lo se puede apreciar que se mejora el rendimiento en dicha propuesta de solución genérica.

Como el comportamiento obtenido para cada una de las funcionalidades probadas fue satisfactorio se puede concluir que: el diseño del clúster de alto rendimiento y alta disponibilidad genérico está validado funcionalmente.

3.7 Conclusiones de las pruebas

CAPÍTULO 3: DISEÑO DE LAS PRUEBAS Y ANÁLISIS DE LOS RESULTADOS

Después de haber realizado todas las pruebas necesarias para validar la propuesta de solución genérica se puede concluir que:

1- Las pruebas realizadas mostraron que se cumplió el objetivo de la investigación y se contrastó la idea a defender en la misma.

Conclusiones del capítulo

A partir de la investigación realizada para obtener el diseño de una propuesta de solución genérica, que permita aumentar la capacidad de respuesta y la disponibilidad en los servidores de bases de datos PostgreSQL, se concluyó que:

1. La implementación de la propuesta de solución basada en un clúster de alto rendimiento y alta disponibilidad se realizaría utilizando la combinación de la herramienta balanceadora de carga *Pgpool-II* y Heartbeat para implementar la alta disponibilidad.
2. Las pruebas realizadas al diseño de solución basadas en este tipo de clúster son necesarias para su validación y ayudan a su perfeccionamiento.

CONCLUSIONES

A partir de la investigación realizada, para obtener el diseño de una propuesta de solución que permite aumentar la capacidad de respuesta y la disponibilidad en sistemas que utilicen medianas y pequeñas bases de datos, se concluyó que:

1. La implementación de la solución basada en un clúster de alta disponibilidad y alto rendimiento, se realizaría utilizando la combinación de herramientas libres como lo son *PgPool*-II para balancear las cargas y *Heartbeat* para implementar la alta disponibilidad.
2. Se le aplicó además pruebas de *benchmark* a dicha propuesta teniendo en cuenta los tiempos de respuestas obtenidos durante la variación de la concurrencia de usuarios y la variación de los nodos del clúster.
3. Las pruebas a los diseños de soluciones basadas en clúster de balanceo de carga son necesarias para su validación y ayudan a su perfeccionamiento.

RECOMENDACIONES

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

1- Seguir el estudio de la solución y mejorarla mediante la implementación de la misma con Hearbeat-3.

2- Utilizar en la versión mejorada de la solución una SAM.

REFERENCIAS BIBLIOGRÁFICAS

Apache Software Foundation . 2009. The Apache Jakarta Proyect. [En línea] 2009.

http://jakarta.apache.org/jmeter/.

Barroso, Luiz André. 2003. Web Search For a Planet:The Google Clúster. *Web Search For a Planet:The Google Clúster.* [En línea] 2003.

http://labs.google.com/papers/googlecluster-ieee.pdf.

Clavijo, Paulo. 2010. Clusters de Alta Disponibilidad (HA). [En línea] 2010.

http://www.lintips.com/?q=node/119.

—. 2010. Soluciones Open Source de Clustering HA. [En línea] 2010.

http://lintips.com/?q=node/120.

CYBERTEC. 2009. CYBERTEC. [En línea] 2009.

http://www.cybertec.at/en/postgresql_products/cybercluster.

González, César. 2006. Iptables en 5 minutos. [En línea] 2006.

http://www.linuca.org/body.phtml?nIdNoticia=99.

Group, PgPool Global Development. 2008. What is pgpool-II. [En línea] 2008.

http://pgpool.projects.postgresql.org/pgpool-II/doc/pgpool-en.html.

HowtoForge. 2010. Installation And Setup Guide For DRBD, OpenAIS, Pacemaker + Xen On OpenSUSE 11.1. [En línea] 2010.

http://www.howtoforge.com/installation-and-setup-guide-for-drbd-openais-pacemaker-xen-on-opensuse-11.1.

Jaume Sabater. 2008. Replicación y alta disponibilidad de PostgreSQL con pgpool-II. [En línea] 2008.

http://linuxsilo.net/articles/postgresql-pgpool.html.

Linux-HA. 2008. Getting Started with Linux-HA (Heartbeat). [En línea] 2008.

http://www.linux-ha.org/GettingStarted.

—. 2006. Heartbeat. [En línea] 2006. http://linux-ha.org/wiki/Heartbeat.

Neustadt, Wiener. 2008. CyberCluster 1.0 Technical Documentation. [En línea] 2008. http://www.cybertec.at/download/dokumentation/documentation_cybercluster.pdf.

Niclausse, Nicolas. 2008. Tsung. [En línea] 2008.

http://tsung.erlang-projects.org/.

Paulo Clavijo. 2010. Introducción a la Alta Disponibilidad. [En línea] 2010. http://www.lintips.com/?q=node/118.

PostgreSQL 9.0beta1 Documentation. 2008. Pgbench. [En línea] 2008. http://developer.postgresql.org/pgdocs/postgres/pgbench.html.

Saz, Ana del. 2008. Configure pgcluster with two nodes. [En línea] 2008.

http://www.mail-archive.com/pgcluster-general@pgfoundry.org/msg00243.html.

Suárez, José. 2002. Sistema HA bajo Linux. [En línea] 2002.

http://www.goa.es/docs/linux_ha_apache.pdf.

The PostgreSQL Conference. 2007. PGCluster-II. [En línea] 2007.

http://www.pgcon.org/2007/schedule/events/6.en.html.

BIBLIOGRAFÍA

Apache Software Foundation . 2009. The Apache Jakarta Proyect. [En línea] 2009. http://jakarta.apache.org/jmeter/.

Barroso, Luiz André. 2003. Web Search For a Planet:The Google Clúster. *Web Search For a Planet:The Google Clúster.* [En línea] 2003.

http://labs.google.com/papers/googlecluster-ieee.pdf.

Brash, Ron. 2008. High Availability Linux - HA On CentOS Howto. [En línea] 2008. http://www.orangespike.ca/content/high-availability-linux-ha-centos-howto.

Cáceres, Juan Esteban. 2007. *Implementación de un servidor Web Apache.* 2007.

Chávez, Dayrel Almaguer. 2008. *Cluster de servidores de bases de datos para aplicaciones web, sobre software libre.* 2008.

—. 2010. Soluciones Open Source de Clustering HA. [En línea] 2010.

http://lintips.com/?q=node/120.

CYBERTEC. 2009. CYBERTEC. [En línea] 2009.

http://www.cybertec.at/en/postgresql_products/cybercluster.

González, César. 2006. Iptables en 5 minutos. [En línea] 2006.

http://www.linuca.org/body.phtml?nIdNoticia=99.

Group, PgPool Global Development. 2008. What is pgpool-II. [En línea] 2008.

http://pgpool.projects.postgresql.org/pgpool-II/doc/pgpool-en.html.

HowtoForge. 2010. Installation And Setup Guide For DRBD, OpenAIS, Pacemaker + Xen On OpenSUSE 11.1. [En línea] 2010.

http://www.howtoforge.com/installation-and-setup-guide-for-drbd-openais-pacemaker-xen-on-opensuse-11.1.

Linux-HA. 2008. Getting Started with Linux-HA (Heartbeat). [En línea] 2008.

http://www.linux-ha.org/GettingStarted.

—. 2006. Heartbeat. [En línea] 2006. http://linux-ha.org/wiki/Heartbeat.

Montero, Adrian Misael Peña. 2009. *Cluster de servidores web para aplicaciones desarrolladas sobre software libre que soportan altos niveles de concurrencia.* 2009.

—. 2008. *Cluster de servidores web para aplicaciones desarrolladas sobre software libre que soportan altos niveles de concurrencia.* 2008.

Niclausse, Nicolas. 2008. Tsung. [En línea] 2008.

http://tsung.erlang-projects.org/.

Suárez, José. 2002. Sistema HA bajo Linux. [En línea] 2002.

http://www.goa.es/docs/linux_ha_apache.pdf.

The PostgreSQL Conference. 2007. PGCluster-II. [En línea] 2007.

http://www.pgcon.org/2007/schedule/events/6.en.html.

GLOSARIO DE TÉRMINOS

UCI: Universidad de las Ciencias Informáticas.

DATEC: Centro de Tecnología de Datos.

SGBD: Sistema Gestor de Bases de Datos.

Clúster: 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.

Supercomputadoras: ordenador con capacidades de cálculo muy superiores a las comunes.

High Performance: Alto rendimiento.

RAC: *Real Application* Clúster*.*

TAF: Recuperación transparente ante fallas.

BSD: *Berkeley Software Distribution.*

ACID: *Atomicity, Consistency, Isolation, Durability.*

MVCC: Control concurrente multiversión.

PITR: Punto en tiempo de recuperación.

Erlang: Lenguaje diseñado para desarrollar sistemas altamente concurrentes y tolerantes a fallos.

SSH: *Secure Shell*, protocolo informático que sirve para acceder a máquinas remotas

RAM: *Random Access Memory*, tipo de memoria de ordenador a la que se puede acceder aleatoriamente.

API: *Application Programming Interface.*

HA: Alta Disponibilidad.

ANEXOS

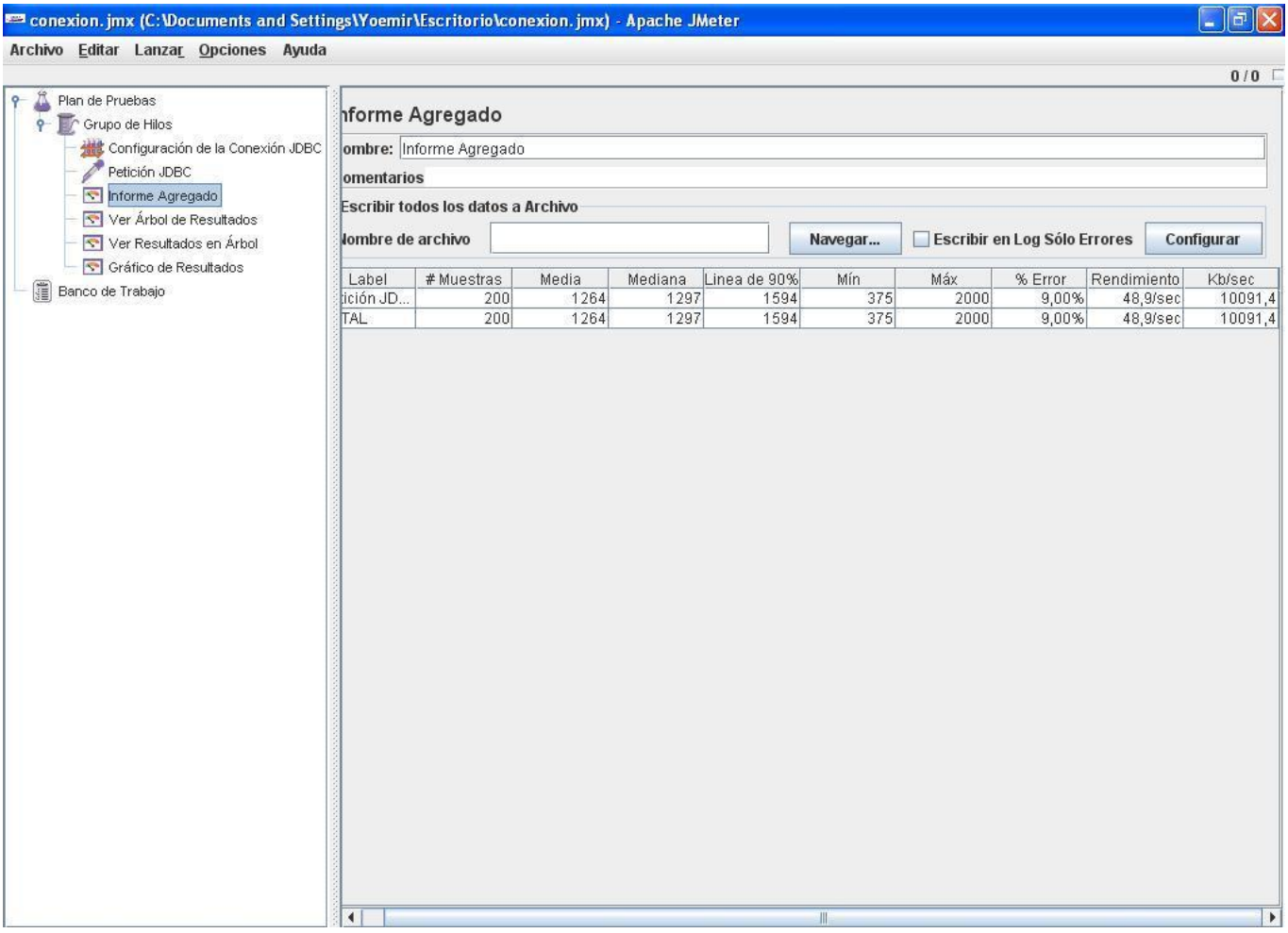
1.1 Método para ver como se balancean las consultas en cada nodo

```
#!/bin/sh
PGSQL=/usr/bin/psql
HEAD=/usr/bin/head
TAIL=/usr/bin/tail
CUT=/usr/bin/cut
IP_LIST="10.34.17.56"
PORT=9999

for ip in $IP_LIST
do
echo "ip address: $ip"
for t in pgbench_branches pgbench_tellers pgbench_accounts pgbench_history
do
echo -n "table $t: "
COUNT=`$PGSQL -h $ip -p $PORT -U pgpool2 -d bench_replication -c "SELECT  count(*)
FROM $t" $
echo $COUNT
done
done

exit 0
```

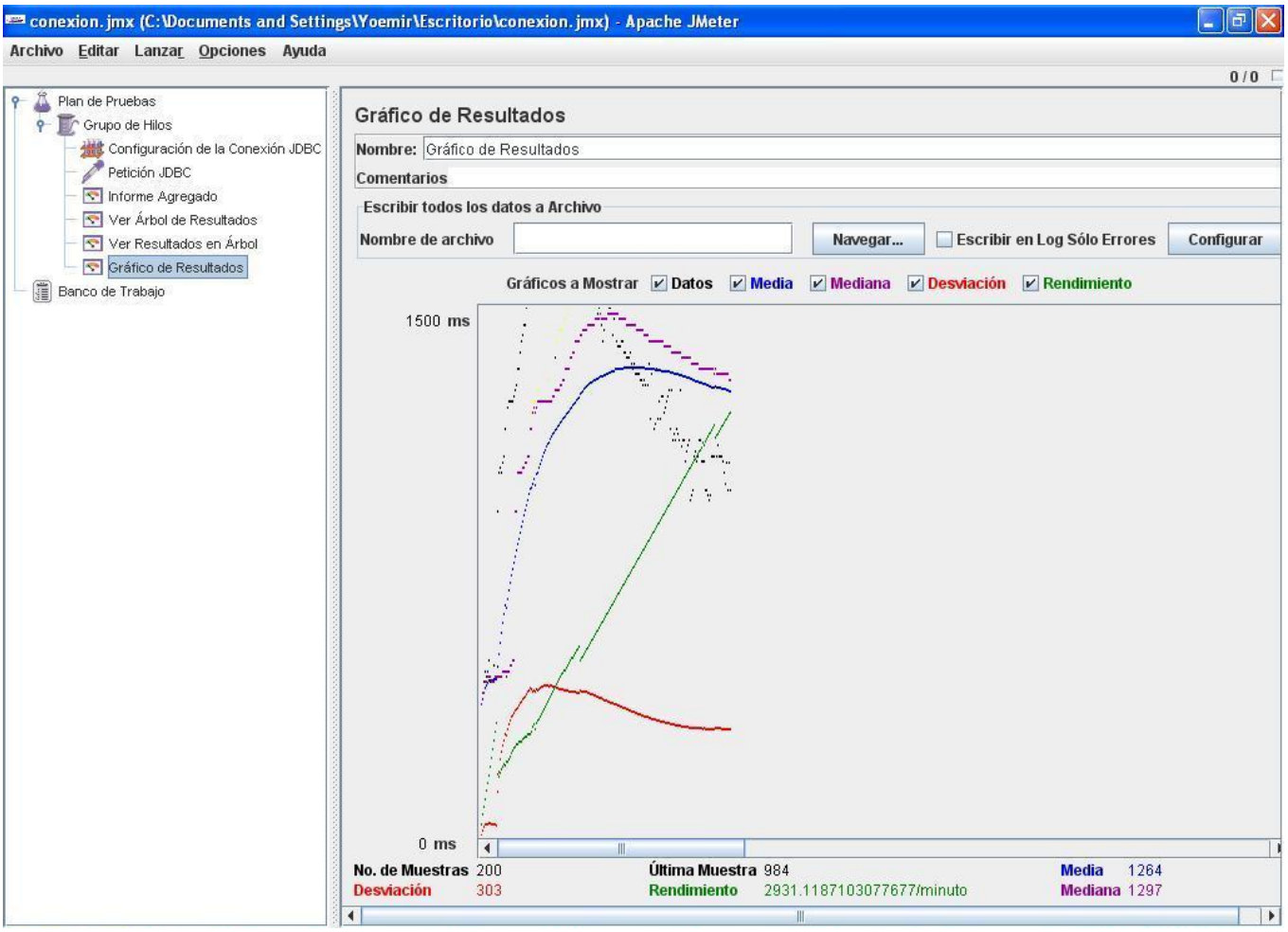
1.2 Método para ver el número de registros insertados en cada nodo

```
#!/bin/sh
PGSQL=/usr/bin/psql
HEAD=/usr/bin/head
TAIL=/usr/bin/tail
CUT=/usr/bin/cut
IP_LIST="10.34.17.55 10.34.17.56"
PORT=5432

for ip in $IP_LIST
do
echo "ip address: $ip"
 for t in pgbench_branches pgbench_tellers pgbench_accounts pgbench_history
 do
echo -n "table $t: "
COUNT=`$PGSQL -h $ip -p $PORT -U pgpool2 -d bench_replication -c "SELECT  count(*)
FROM $t" $
echo $COUNT
done
done

exit 0
```

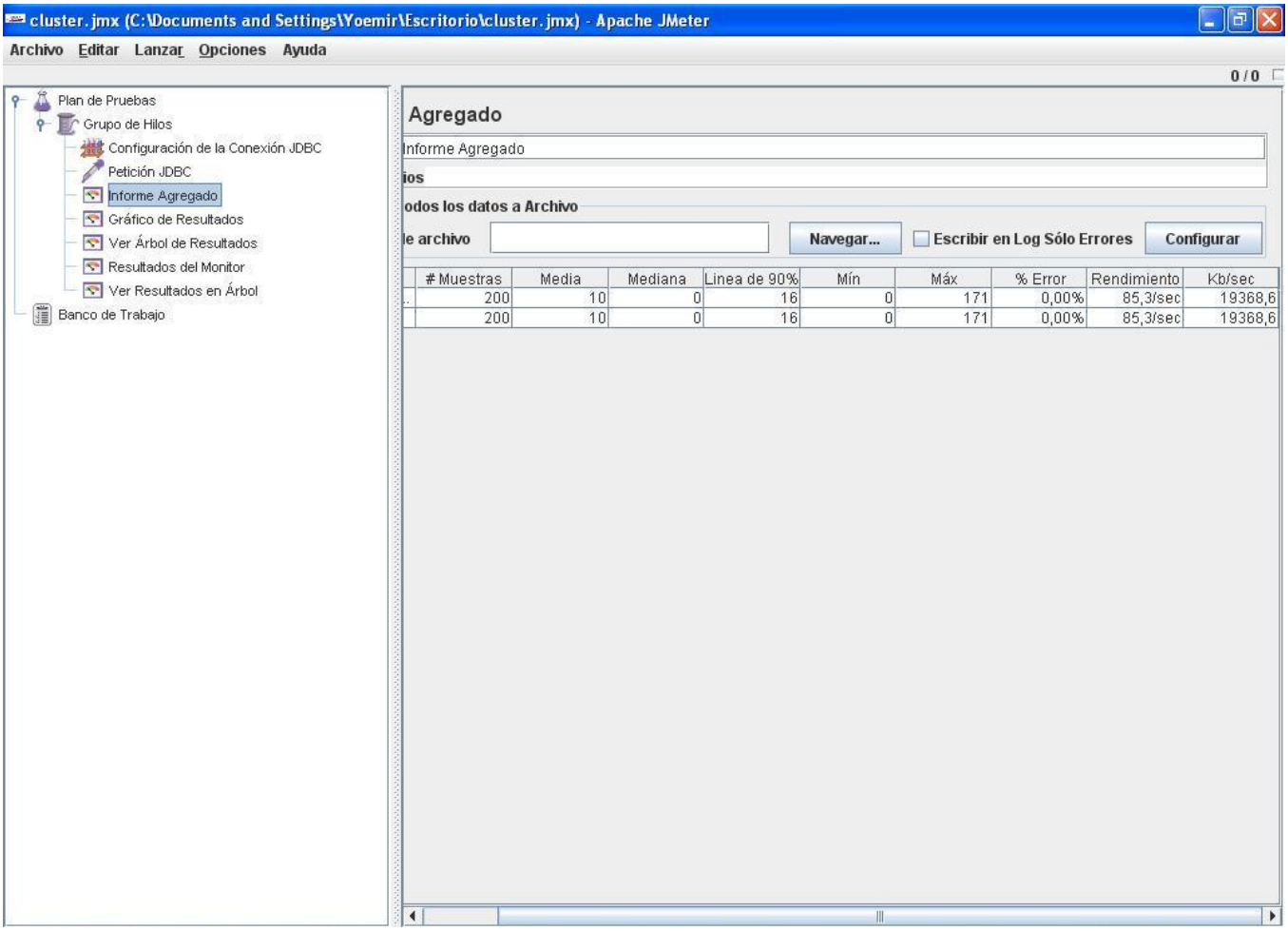
1.3 Informe agregado de las pruebas con *Jmeter* al servidor PostgreSQL solamente.



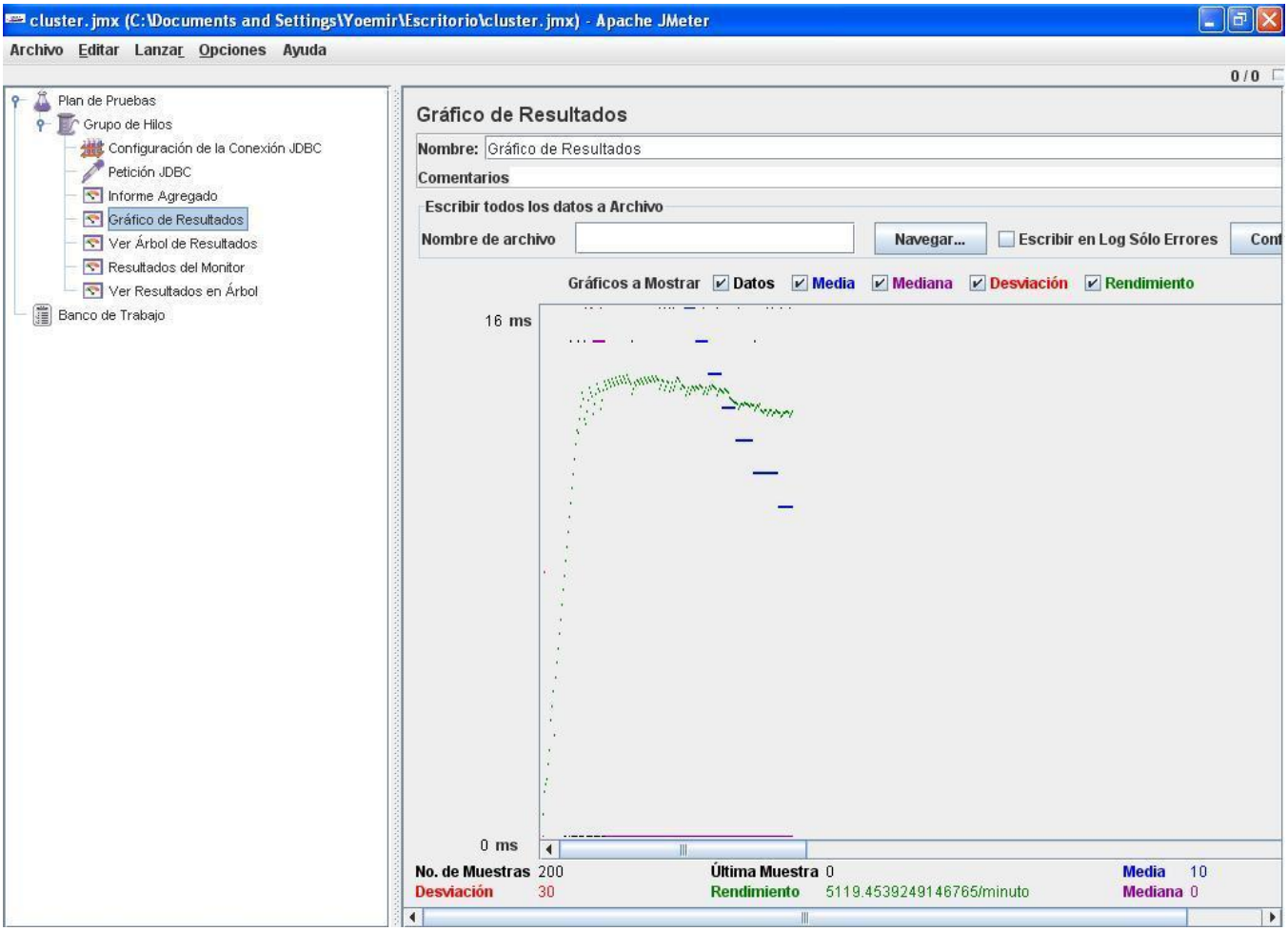
1.4 Gráfico de resultados de las pruebas con *Jmeter* al servidor PostgreSQL solamente.



1.5 Informe agregado de las pruebas con Jmeter al clúster.



1.6 Gráfico de resultados de las pruebas con *Jmeter* al clúster.



1.7 Configuración de Iptables para PgPool-II

```
#!/bin/sh

## Ejemplo de script para proteger la propia máquina

echo -n Aplicando Reglas de Firewall...


## FLUSH de reglas

iptables -F

iptables -X

iptables -Z

iptables -t nat –F


## Establecemos politica por defecto

iptables -P INPUT ACCEPT

iptables -P OUTPUT ACCEPT

iptables -P FORWARD ACCEPT

iptables -t nat -P PREROUTING ACCEPT

iptables -t nat -P POSTROUTING ACCEPT


## Empezamos a filtrar

# El localhost se deja (por ejemplo conexiones locales a mysql)

iptables -A INPUT -i lo -j ACCEPT
```

```
# A nuestra IP le dejamos todo

iptables –A INPUT –s 10.34.17.55 –j ACCEPT

iptables –A INPUT –s 10.34.17.56 –j ACCEPT

iptables –A INPUT –s 10.34.17.180 –j ACCEPT


# Abrimos los puertos deseados

iptables –A INPUT –p tcp -–dport 5432 –j ACCEPT

iptables –A INPUT –p tcp –-dport 22 –j ACCEPT

iptables –A INPUT –p tcp –-dport 694 –j ACCEPT

iptables –A INPUT –p tcp –-dport 9898 –j ACCEPT

iptables –A INPUT –p tcp –-dport 9999 –j ACCEPT

echo “ OK . Verifique que lo que se aplica con: iptables –L -n“

# Fin del script
```