UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

IMPLEMENTACIÓN DE UN CLUSTER DE ALTA DISPONIBILIDAD DE BASE DE DATOS PARA LA GEDGAPA

T E S I S

QUE PARA OBTENER EL TÍTULO DE INGENIERO EN COMPUTACIÓN

PRESENTA:

DENISSE HAIDEE IBAÑEZ LUNA

DIRECTOR DE TESIS:

DR. VÍCTOR HUGO JACOBO ARMENDÁRIZ

CIUDAD UNIVERSITARIA **2009**

Agradecimientos

Agradezco a Dios por tener la oportunidad de permitirme finalizar mis estudios y por darme todo lo que tengo en la vida.

Agradezco con todo el corazón a mis padres porque a ellos les debo todo lo que soy y la persona en la que me he convertido a través de los años.

A mi padre que sé que desde el maravilloso lugar donde está, siempre me cuidó y deseó lo mejor para mí y sé que sin duda se hubiera sentido orgulloso de verme justo en el lugar en el que estoy, gracias papá por llenar mi infancia de amor.

A mi mami, por ser esa única persona, que siempre y a pesar de todo siempre esta a mi lado, todo lo que soy te lo debo a ti y a tú gran esfuerzo, espero algún día llegar hacer esa gran mujer que eres tú, te quiero mami mil gracias por todo.

A toda mi familia y amigos por brindarme su cariño y apoyo incondicional.

A todo el personal de la DGAPA, al Dr. Víctor Hugo Jacobo Armendáriz por fungir como mi tutor de tesis y tomarse el tiempo para asesorarme y encaminarme en lo reflejado en este trabajo de tesis, también agradezco especialmente al Ing. Arturo Bahena Armas por permitirme realizar este proyecto en su área y brindarme su asesoría durante todo el transcurso del mismo. Al Ing. Felipe Ramos por tener la paciencia y disposición de transmitirme sus conocimientos y sobre todo por ser un gran compañero y un ejemplo a seguir. También un agradecimiento al Ing. Alejandro Dávila por contribuir al proyecto y por darle el toque divertido al trabajo y por último al Ing. Jorge Arturo González por la motivación brindada en la realización del proyecto, muchas gracias a todos por enseñarme lo que significa el trabajo en equipo.

Pero sobre todo agradezco a la mejor Universidad de México, la UNAM por permitirme formar parte de su comunidad y pasar en ella los mejores años de mi vida, Gracias.

Denisse Haidee

ÍNDICE

Capítulo 1 Antecedentes 2

1.1 Introducción 2

1.2 Dirección General de Asuntos del Personal Académico (DGAPA) 2

1.3 Definición del problema 5

1.4 Objetivo 6

1.5 Definición del proyecto 8

Capítulo 2 Marco Teórico 11

2.1 Introducción 11

2.2 Requerimientos de un sistema 11

2.3 Cluster 14

2.3.1 Orígenes y un poco de historia 14

2.3.2 Funcionamiento de un cluster 16

2.3.3 *Downtime* 18

2.3.4 Beneficios de un Cluster 20

2.3.5 Tipos de clusters 21

2.3.6 Componentes de un Cluster 24

2.3.7 Arquitectura de un cluster 26

2.4 Replicación con PostgreSQL 34

2.4.1 Bases de datos relacionales 34

2.4.2 SQL 37

2.4.3 PostgreSQL 38

2.4.4 Replicación 40

Capítulo 3 Descripción del sistema 50

3.1 Introducción 50

3.2 Requerimientos funcionales del cluster 50

3.2.1 Planeación de un cluster 50

3.3 Modelo del análisis 52

3.3.1 Slony-I 52

3.3.2 Seguridad 61

3.3.3 Sincronización del tiempo 64

3.3.4 Migración 67

3.3.5 Consola de administración del cluster 70

3.3.6 Pgpool-II 73

3.3.7 Monitoreo 75

3.3.8 *Hardware* del cluster 83

3.3.9 *Software* del cluster 83

3.4 Diseño 84

3.4.1 Diseño físico del cluster 84

3.4.2 Diseño de los *sets* de replicación de Slony-I 85

3.4.3 Diseño de sincronización 87

3.4.4. Mecanismo de respaldo 88

3.4.5 Diseño del proceso de migración 90

3.4.6 Diseño de las herramientas de monitoreo 91

3.4.7 Diseño de la implementación del cluster 92

Capítulo 4 Implementación y pruebas 95

4.1 Introducción 95

4.2 Instalación y configuración del cluster 95

4.2.1 Instalación y configuración del sistema operativo 95

4.2.2 Instalación y configuración de la seguridad 96

4.2.3 Configuración de la sincronización del tiempo 99

4.2.4 Instalación y configuración de PostgreSQL 101

4.2.5 Proceso de respaldo 107

4.2.6 Migración 109

4.2.7 Instalación y configuración de Slony-I 117

4.2.8 Instalación y configuración de Pgpool-II 125

4.2.9 Pruebas 132

4.2.10 Monitoreo 141

Conclusiones 152

Bibliografía 155

Mesografía 155

Apéndices 157

A.1 Glosario 157

A.2 Diagrama de Gantt 168

Capítulo 1
Antecedentes

Capítulo 1 Antecedentes

1.1 Introducción

En el primer capítulo, se profundiza sobre los antecedentes históricos de la fundación de la Dirección General de Asuntos del Personal Académico (DGAPA),la descripción de algunas de las funciones que realiza la dependencia dentro de la Universidad en beneficio de la comunidad universitaria.

Se presenta también una breve introducción de los programas que son responsabilidad de la DGAPA y que buscan fomentar la superación del personal acádemico.

Finalmente se aborda la problemática que presenta la dependencia en el área de la gestión electrónica en sus siglas, geDGAPA, describiendo su funcionamiento y estructura actual, así como las posibles alternativas de solución que se plantean para resolver dicha problemática e implementar la solución más óptima.

1.2 Dirección General de Asuntos del Personal Académico (DGAPA)

La Dirección General de Asuntos del Personal Académico (DGAPA) fue fundada en el año de 1977 con el fin de promover la superación del personal académico de la Universidad Nacional Autónoma de México.[1]

La DGAPA impulsa la superación de los académicos apoyándolos para realizar estudios de posgrado, llevar a cabo estancias sabáticas o de investigación, tanto en el extranjero como en diversas instituciones de educación superior del país; asimismo, fortalece la formación de recursos humanos de alto nivel otorgando becas para realizar estancias posdoctorales en entidades académicas de la propia UNAM.

Respecto a la investigación, la DGAPA favorece su ejercicio apoyando a las dependencias, a los investigadores, a los profesores y a los técnicos académicos de todos los niveles de nuestra máxima Casa de Estudios. Los proyectos de investigación que año con año la dirección apoya, gozan de reconocido prestigio tanto a nivel nacional como internacional, de tal modo que es difícil concebir la vida académica de nuestra Universidad sin tomar en cuenta los apoyos que se otorgan a los programas que esta dependencia administra.

Con relación a la docencia, se fomenta la participación de los académicos en los cursos de actualización y en los distintos diplomados dirigidos a profesores del bachillerato y de licenciatura; asimismo, se administran proyectos de investigación con el objeto de promover en la UNAM la innovación y el mejoramiento de la enseñanza-aprendizaje.

Tiene también, entre sus funciones, las de administrar los programas de estímulos y organizar los reconocimientos que la Universidad otorga a sus académicos distinguidos, a la vez

[1] http://dgapa.unam.mx

que se ocupa de los asuntos relacionados con aquellos que han sido designados como profesores e investigadores eméritos.

En conclusión, la DGAPA a lo largo de sus más de 30 años ha incrementado el número de los programas que administra para impulsar la vida académica, de tal modo que hasta la fecha, contribuye de manera notable en el cumplimiento de las funciones sustantivas de la UNAM estipuladas en su Estatuto General de *"formar profesionistas, investigadores, profesores universitarios y técnicos útiles a la sociedad; organizar y realizar investigaciones principalmente acerca de las condiciones y problemas nacionales, y extender con la mayor amplitud posible los beneficios de la cultura."*

Con el objetivo de informar como se encuentra organizada la DGAPA, a continuación se muestra el organigrama de la dependencia en la figura 1.1.[2]

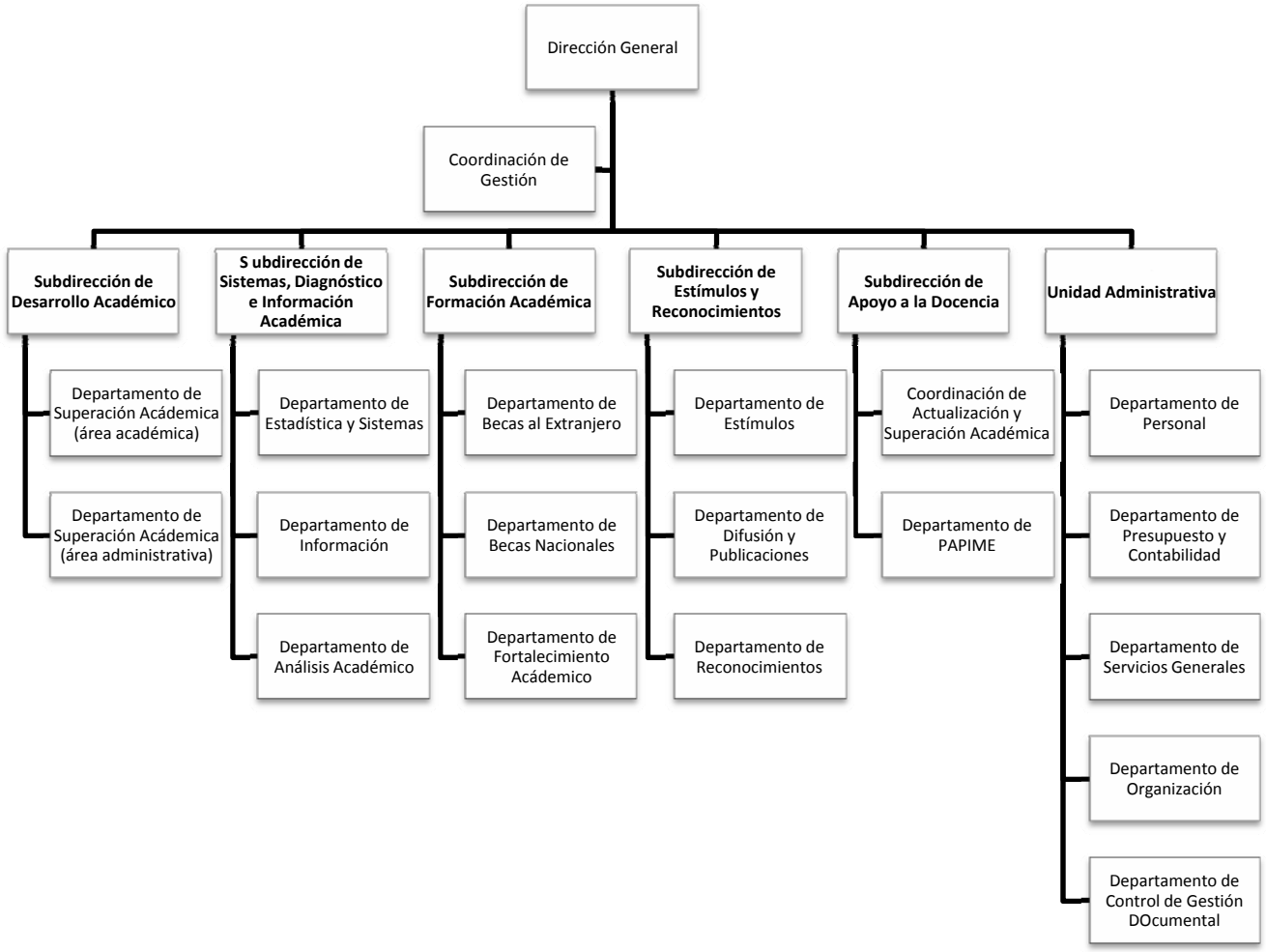


Figura 1.1 Organigrama de la DGAPA

[2] http://dgapa.unam.mx/org_nuevo09/orgral08.html

Debido a su carácter de impulsora de la carrera profesional del personal académico de la UNAM, se le ha asignado las tareas de coordinar programas relacionados con las actividades del personal académico, cabe mencionar algunos de los más importantes:

Programa de Apoyos para la Superación del Personal Académico de la UNAM (PASPA)

El PASPA contribuye a la superación del personal académico y al fortalecimiento de la planta académica de las entidades, mediante apoyos para realizar estudios de posgrado o estancias sabáticas, posdoctorales y de investigación.[3]

Programa de Apoyo a Proyectos para la Innovación y Mejoramiento de la Enseñanza (PAPIME)

El PAPIME impulsa la superación y desarrollo del personal académico con el apoyo a proyectos que conduzcan a la innovación y al mejoramiento de la enseñanza en el bachillerato y la licenciatura.[4]

Iniciativa para Fortalecer la Carrera Académica en el Bachillerato de la UNAM (INFOCAB)

El INFOCAB tiene como objetivo promover la participación de los profesores en actividades académicas con un programa integral de apoyo y fortalecimiento de su carrera académica sustentado en la vida académica cotidiana del bachillerato (ENP y CCH). [5]

Programa de Actualización y Superación Docente (PASD)

El PASD tiene como objetivo promover y apoyar la actualización y superación de los profesores de licenciatura y bachillerato en el área que imparten sus clases, tanto en lo que se refiere a los contenidos, como a las técnicas y tendencias didácticas.[6]

Programa de Apoyo a Proyectos de Investigación e Innovación Tecnológica (PAPIIT)

Con el PAPIIT se busca impulsar el desarrollo de proyectos de investigación básica, aplicada y multidisciplinaria de alta calidad en las áreas del conocimiento que se llevan a cabo en facultades, escuelas, institutos y centros, que fomenten la formación de cuadros de investigación. [7]

Premio Universidad Nacional (PUN)

El PUN tiene como objetivo reconocer a los profesores, investigadores o técnicos académicos, que se han destacado en el cumplimiento de las funciones sustantivas de docencia, investigación y difusión de la cultura. [8]

[3] http://dgapa.unam.mx/programas/f_paspa/paspa.html
[4] http://dgapa.unam.mx/programas/a_papime/papime.html
[5] http://dgapa.unam.mx/programas/a_infocab/infocab.html
[6] http://dgapa.unam.mx/programas/a_pasd/pasd.html
[7] http://dgapa.unam.mx/programas/d_papiit/papiit.html
[8] http://dgapa.unam.mx/programas/r_pun/pun.html

Reconocimiento Distinción Universidad Nacional para Jóvenes Académicos (RDUNJA)

El RDUNJA tiene como objetivo fomentar y promover el potencial de los jóvenes académicos y estimular sus esfuerzos por la superación constante de su trabajo.[9]

Programa de Estímulos a la Productividad y al Rendimiento del Personal Académico de Asignatura (PEPASIG)

El PEPASIG tiene como objetivo estimular la labor de los profesores de asignatura que hayan realizado una labor sobresaliente, así como, elevar el nivel de productividad y calidad del desempeño académico.[10]

Programa de Primas al Desempeño del Personal Académico de Tiempo Completo (PRIDE)

El PRIDE tiene como objetivo estimular la labor del personal académico que haya realizado una labor sobresaliente, así como elevar el nivel de productividad y calidad del desempeño del personal académico.[11]

Con el manejo de los programas mencionados anteriormente la DGAPA fortalece y fomenta la superación del personal académico de la Universidad.

1.3 Definición del problema

La Dirección General de Asuntos de Personal Académico (DGAPA) siempre pendiente de ofrecer un mejor servicio, requiere de un sistema de alta disponibilidad, para mejorar el servicio que se brinda a la comunidad universitaria.

Actualmente la DGAPA proporciona servicios en línea de los programas académicos que administra mediante un sistema modular denominado geDGAPA (gestión electrónica de la DGAPA), el cual está siendo programado en términos generales en Java con acceso a bases de datos del tipo PostgresSQL, el sistema de gestión está soportado por un *cluster* de servidor de aplicaciones del tipo Tomcat, en el cual se distribuyen cada uno de los módulos generados, por el ciclo de vida del software de la dependencia.

Además del *cluster* de Tomcat, la geDGAPA cuenta con un único servidor de PostgreSQL versión 7.4.6, en el cual se almacena toda la información de la gestión electrónica de la dependencia, hoy en día la plataforma soporta 68,049 roles registrados de los cuales 47,530 cuentan con acceso de usuario a la gestión electrónica, es importante mencionar que la geDGAPA tiene 4 años desde su creación y su desarrollo actualmente se estima en un 40% de avance de la totalidad de módulos desarrollados.

Cabe mencionar que cuando se han realizado modificaciones a la configuración del servidor de base de datos, con la finalidad de soportar la carga de usuarios, éste sigue estando paulatinamente rebasado en su capacidad y desempeño (*performance*), debido al número de peticiones que están siendo solicitadas en tiempos críticos de cierre de convocatorias de solicitud

[9] http://dgapa.unam.mx/programas/r_rdunja/rdunja.html
[10] http://dgapa.unam.mx/programas/e_pepasig/pepasig.html
[11] http://dgapa.unam.mx/programas/e_pride/pride.html

PRÓLOGO

En los sistemas de información los datos que se almacenan son parte fundamental y considerados de gran relevancia, por la importancia que representan hoy en día, por ello es de vital importancia tener disponible y segura la información en todo momento.

El trabajo de tesis que aquí se presenta, deriva de esa misma necesidad de disponer de la información dentro de una de las dependencias que brinda servicios fundamentales para el personal académico de la UNAM, la Dirección General de Asuntos de Personal Académico (DGAPA), en su área de gestión electrónica (geDGAPA), es la encargada de proporcionar una plataforma de diversos sistemas dirigidos a los académicos, por lo tanto, es imprescindible brindar un servicio de calidad.

De lo anterior surge la idea de implementar un *cluster* de alta disponibilidad para las bases de datos que contienen información de carácter crítico, que por su importancia es indispensable proteger y resguardar ante alguna contingencia informática que se pueda presentar y arriesgue la información utilizada por los sistemas de la geDGAPA.

En consecuencia, es necesario tener un conjunto de equipos que funcionen como uno solo y sea transparente para el usuario, tal como funciona un *cluster* y brinde alta disponibilidad con la única finalidad de mejorar los servicios que se brindan actualmente.

Capítulo 1
Antecedentes

Capítulo 1 Antecedentes

1.1 Introducción

En el primer capítulo, se profundiza sobre los antecedentes históricos de la fundación de la Dirección General de Asuntos del Personal Académico (DGAPA),la descripción de algunas de las funciones que realiza la dependencia dentro de la Universidad en beneficio de la comunidad universitaria.

Se presenta también una breve introducción de los programas que son responsabilidad de la DGAPA y que buscan fomentar la superación del personal acádemico.

Finalmente se aborda la problemática que presenta la dependencia en el área de la gestión electrónica en sus siglas, geDGAPA, describiendo su funcionamiento y estructura actual, así como las posibles alternativas de solución que se plantean para resolver dicha problemática e implementar la solución más óptima.

1.2 Dirección General de Asuntos del Personal Académico (DGAPA)

La Dirección General de Asuntos del Personal Académico (DGAPA) fue fundada en el año de 1977 con el fin de promover la superación del personal académico de la Universidad Nacional Autónoma de México.[1]

La DGAPA impulsa la superación de los académicos apoyándolos para realizar estudios de posgrado, llevar a cabo estancias sabáticas o de investigación, tanto en el extranjero como en diversas instituciones de educación superior del país; asimismo, fortalece la formación de recursos humanos de alto nivel otorgando becas para realizar estancias posdoctorales en entidades académicas de la propia UNAM.

Respecto a la investigación, la DGAPA favorece su ejercicio apoyando a las dependencias, a los investigadores, a los profesores y a los técnicos académicos de todos los niveles de nuestra máxima Casa de Estudios. Los proyectos de investigación que año con año la dirección apoya, gozan de reconocido prestigio tanto a nivel nacional como internacional, de tal modo que es difícil concebir la vida académica de nuestra Universidad sin tomar en cuenta los apoyos que se otorgan a los programas que esta dependencia administra.

Con relación a la docencia, se fomenta la participación de los académicos en los cursos de actualización y en los distintos diplomados dirigidos a profesores del bachillerato y de licenciatura; asimismo, se administran proyectos de investigación con el objeto de promover en la UNAM la innovación y el mejoramiento de la enseñanza-aprendizaje.

Tiene también, entre sus funciones, las de administrar los programas de estímulos y organizar los reconocimientos que la Universidad otorga a sus académicos distinguidos, a la vez que se ocupa de los asuntos relacionados con aquellos que han sido designados como profesores e investigadores eméritos.

[1] http://dgapa.unam.mx

En conclusión, la DGAPA a lo largo de sus más de 30 años ha incrementado el número de los programas que administra para impulsar la vida académica, de tal modo que hasta la fecha, contribuye de manera notable en el cumplimiento de las funciones sustantivas de la UNAM estipuladas en su Estatuto General de *"formar profesionistas, investigadores, profesores universitarios y técnicos útiles a la sociedad; organizar y realizar investigaciones principalmente acerca de las condiciones y problemas nacionales, y extender con la mayor amplitud posible los beneficios de la cultura."*

Con el objetivo de informar como se encuentra organizada la DGAPA, a continuación se muestra el organigrama de la dependencia en la figura 1.1.[2]

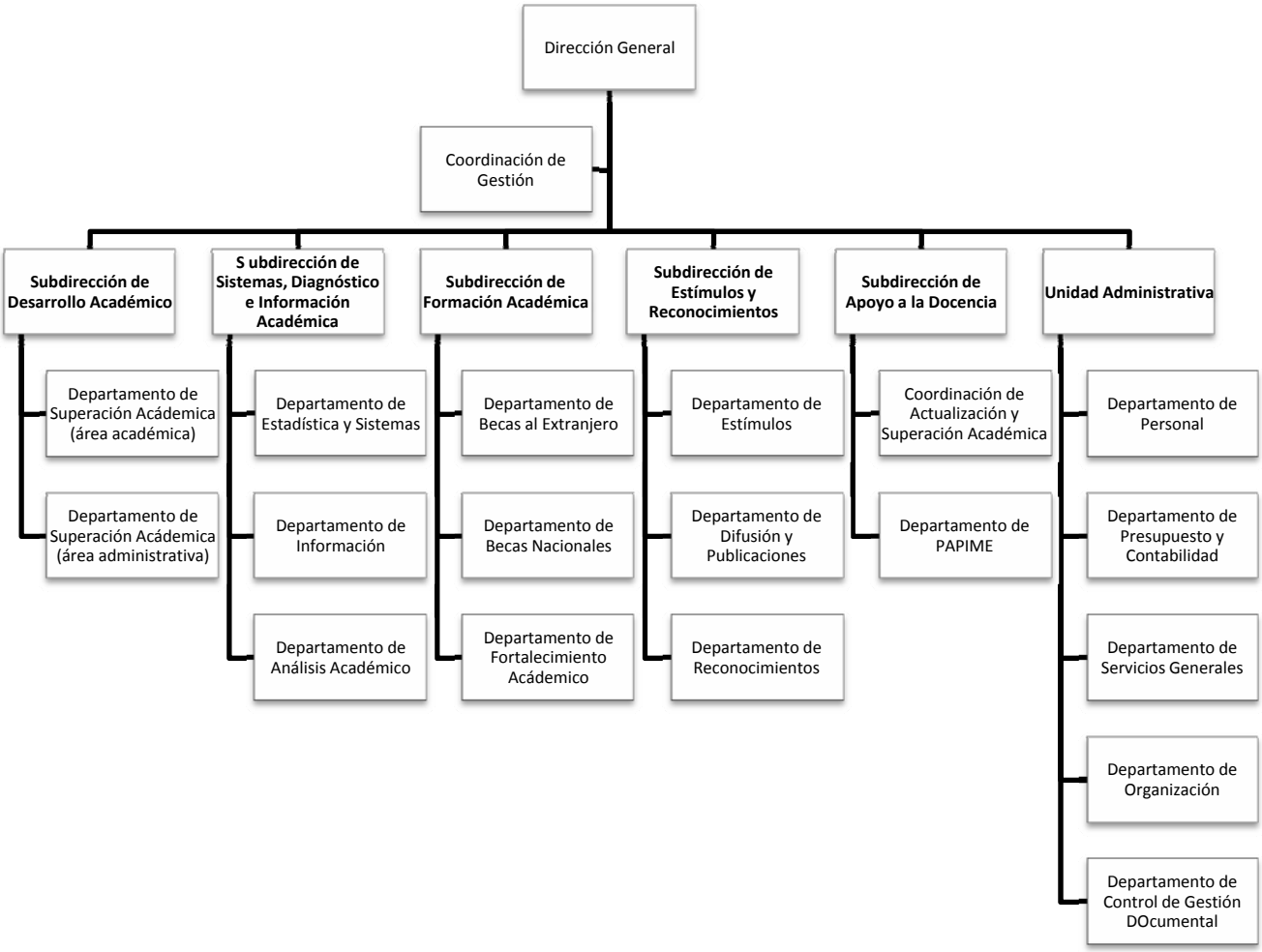


Figura 1.1 Organigrama de la DGAPA

Debido a su carácter de impulsora de la carrera profesional del personal académico de la UNAM, se le ha asignado las tareas de coordinar programas relacionados con las actividades del personal académico, cabe mencionar algunos de los más importantes:

[2] http://dgapa.unam.mx/org_nuevo09/orgral08.html

Programa de Apoyos para la Superación del Personal Académico de la UNAM (PASPA)

El PASPA contribuye a la superación del personal académico y al fortalecimiento de la planta académica de las entidades, mediante apoyos para realizar estudios de posgrado o estancias sabáticas, posdoctorales y de investigación.[3]

Programa de Apoyo a Proyectos para la Innovación y Mejoramiento de la Enseñanza (PAPIME)

El PAPIME impulsa la superación y desarrollo del personal académico con el apoyo a proyectos que conduzcan a la innovación y al mejoramiento de la enseñanza en el bachillerato y la licenciatura.[4]

Iniciativa para Fortalecer la Carrera Académica en el Bachillerato de la UNAM (INFOCAB)

El INFOCAB tiene como objetivo promover la participación de los profesores en actividades académicas con un programa integral de apoyo y fortalecimiento de su carrera académica sustentado en la vida académica cotidiana del bachillerato (ENP y CCH). [5]

Programa de Actualización y Superación Docente (PASD)

El PASD tiene como objetivo promover y apoyar la actualización y superación de los profesores de licenciatura y bachillerato en el área que imparten sus clases, tanto en lo que se refiere a los contenidos, como a las técnicas y tendencias didácticas.[6]

Programa de Apoyo a Proyectos de Investigación e Innovación Tecnológica (PAPIIT)

Con el PAPIIT se busca impulsar el desarrollo de proyectos de investigación básica, aplicada y multidisciplinaria de alta calidad en las áreas del conocimiento que se llevan a cabo en facultades, escuelas, institutos y centros, que fomenten la formación de cuadros de investigación. [7]

Premio Universidad Nacional (PUN)

El PUN tiene como objetivo reconocer a los profesores, investigadores o técnicos académicos, que se han destacado en el cumplimiento de las funciones sustantivas de docencia, investigación y difusión de la cultura. [8]

Reconocimiento Distinción Universidad Nacional para Jóvenes Académicos (RDUNJA)

El RDUNJA tiene como objetivo fomentar y promover el potencial de los jóvenes académicos y estimular sus esfuerzos por la superación constante de su trabajo.[9]

[3] http://dgapa.unam.mx/programas/f_paspa/paspa.html
[4] http://dgapa.unam.mx/programas/a_papime/papime.html
[5] http://dgapa.unam.mx/programas/a_infocab/infocab.html
[6] http://dgapa.unam.mx/programas/a_pasd/pasd.html
[7] http://dgapa.unam.mx/programas/d_papiit/papiit.html
[8] http://dgapa.unam.mx/programas/r_pun/pun.html

Programa de Estímulos a la Productividad y al Rendimiento del Personal Académico de Asignatura (PEPASIG)

El PEPASIG tiene como objetivo estimular la labor de los profesores de asignatura que hayan realizado una labor sobresaliente, así como, elevar el nivel de productividad y calidad del desempeño académico.[10]

Programa de Primas al Desempeño del Personal Académico de Tiempo Completo (PRIDE)

El PRIDE tiene como objetivo estimular la labor del personal académico que haya realizado una labor sobresaliente, así como elevar el nivel de productividad y calidad del desempeño del personal académico.[11]

Con el manejo de los programas mencionados anteriormente la DGAPA fortalece y fomenta la superación del personal académico de la Universidad.

1.3 Definición del problema

La Dirección General de Asuntos de Personal Académico (DGAPA) siempre pendiente de ofrecer un mejor servicio, requiere de un sistema de alta disponibilidad, para mejorar el servicio que se brinda a la comunidad universitaria.

Actualmente la DGAPA proporciona servicios en línea de los programas académicos que administra mediante un sistema modular denominado geDGAPA (gestión electrónica de la DGAPA), el cual está siendo programado en términos generales en Java con acceso a bases de datos del tipo PostgresSQL, el sistema de gestión está soportado por un *cluster* de servidor de aplicaciones del tipo Tomcat, en el cual se distribuyen cada uno de los módulos generados, por el ciclo de vida del software de la dependencia.

Además del *cluster* de Tomcat, la geDGAPA cuenta con un único servidor de PostgreSQL versión 7.4.6, en el cual se almacena toda la información de la gestión electrónica de la dependencia, hoy en día la plataforma soporta 68,049 roles registrados de los cuales 47,530 cuentan con acceso de usuario a la gestión electrónica, es importante mencionar que la geDGAPA tiene 4 años desde su creación y su desarrollo actualmente se estima en un 40% de avance de la totalidad de módulos desarrollados.

Cabe mencionar que cuando se han realizado modificaciones a la configuración del servidor de base de datos, con la finalidad de soportar la carga de usuarios, éste sigue estando paulatinamente rebasado en su capacidad y desempeño (*performance*), debido al número de peticiones que están siendo solicitadas en tiempos críticos de cierre de convocatorias de solicitud de información, por lo que es de esperar que el número de usuarios vaya creciendo de acuerdo al desarrollo de la geDGAPA y que el servidor de PostgreSQL ya no pueda soportar el número de usuarios y peticiones hacia él.

[9] http://dgapa.unam.mx/programas/r_rdunja/rdunja.html
[10] http://dgapa.unam.mx/programas/e_pepasig/pepasig.html
[11] http://dgapa.unam.mx/programas/e_pride/pride.html

Por lo tanto se plantea utilizar la tecnología de *cluster* en el servidor de bases de datos PostgreSQL para incrementar su potencia y disponibilidad con el fin de contar con tecnología de *cluster* de bases de datos adecuada (*ad-hoc*) al servidor de aplicaciones para soportar la carga actual y futura en geDGAPA, asimismo dentro de este trabajo se realizará la actualización de la versión del servidor de PostgreSQL, así como, la realización de la migración de la información, pruebas y monitoreo del funcionamiento del nuevo *cluster*.

En la figura 1.2 se muestra la estructura actual de la geDGAPA, donde interactúa el servidor web con el servidor de aplicaciones Tomcat que se comunica con el único servidor de PostgreSQL para responder a las peticiones de los clientes.

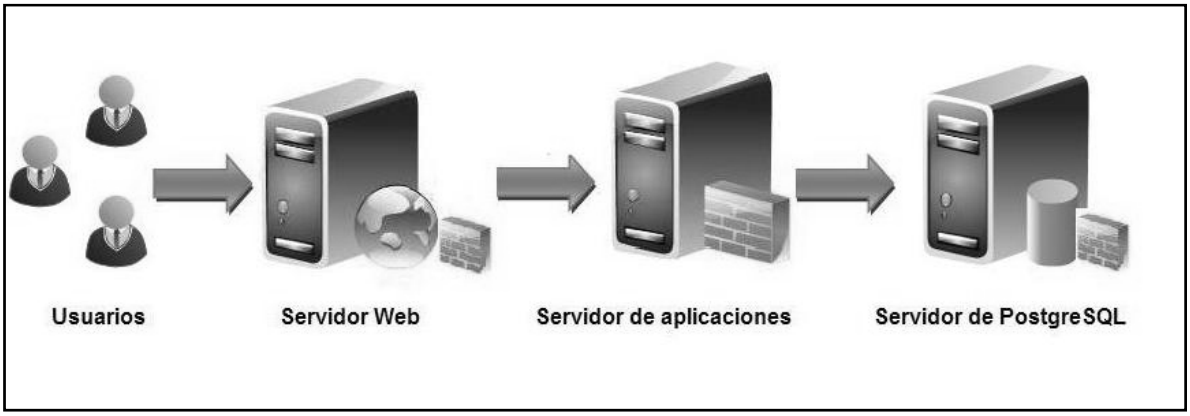


Figura 1.2 Estructura actual de la geDGAPA.

Al establecer los sistemas de información basados en computadoras se debe tener la certeza de que se logren dos objetivos principales: que sea un sistema correcto y que este correcto el sistema. Ningún sistema que deje de satisfacer ambos objetivos será completamente útil para la organización.

1.4 Objetivo

En la actualidad para muchas organizaciones, los sistemas de información basados en computadoras son el corazón de las actividades cotidianas y objeto de gran consideración. A medida que las aplicaciones crecen y son más críticas, la alta disponibilidad de los servicios se vuelve más importante.

Una de las ventajas de un *cluster* es que posee redundancia de *hardware* y de *software*. La alta disponibilidad puede ser provista mediante la detección de la falla de un nodo o servicio y reconfigurando el sistema apropiadamente para que la carga de trabajo pueda ser manejada por los restantes nodos del *cluster.*

Con la tecnología *cluster*, se conserva la disponibilidad de la información, porque se utiliza una replicación asíncrona con una arquitectura maestro-esclavo, en consecuencia, mientras el maestro replica en un esclavo, los datos a los que se tendrá acceso siempre serán a los del maestro por ser los más actuales, mientras los demás nodos esclavos se replican, el nodo maestro

estará disponible para consultar los datos y en caso de que llegará a fallar, el nodo esclavo mas actualizado antes de la ocurrencia de la falla tomará el lugar del nodo maestro para responder a las peticiones de los clientes, por lo tanto en todo momento habrá disponibilidad de la información y con un tiempo de respuesta oportuno.

Además la utilización de un *cluster* permite la escalabilidad, lo que se puede entender como, la capacidad de modificar el sistema de manera conveniente a las crecientes necesidades que puedan surgir en un período de tiempo, es decir, que es un sistema que va a estar vigente en un futuro.

Por lo anterior la implementación de un *cluster* de alta disponibilidad de bases de datos para la geDGAPA tiene como objetivos analizar, diseñar, desarrollar e implementar un sistema de base de datos basado en tecnología *cluster* que contribuya al mejoramiento de los servicios en línea que proporciona la geDGAPA a la comunidad universitaria, a fin de brindar a los usuarios una herramienta de gestión electrónica ágil y eficaz.

En la figura 1.3 se observa un diagrama donde se observa las ventajas que brindará el *cluster* al interactuar con los usuarios y que deriva a su vez en un mejor servicio brindado por la gestión electrónica de la DGAPA.

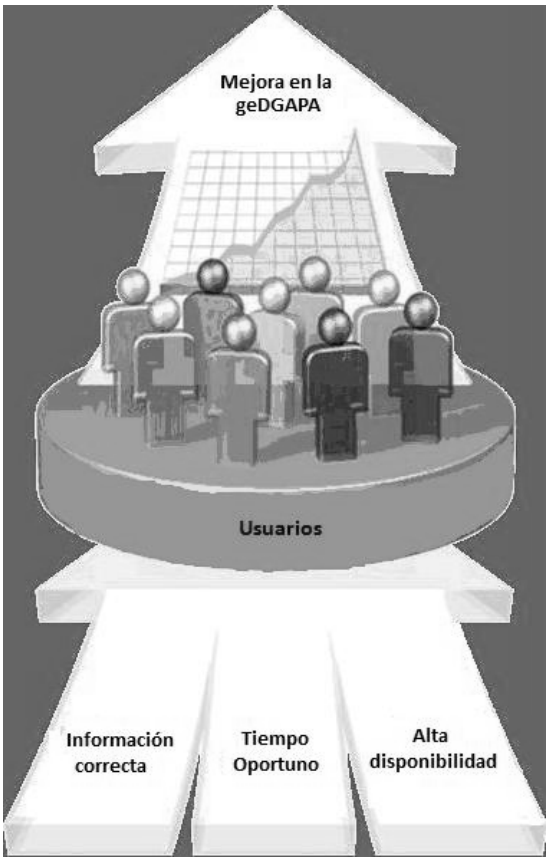


Figura 1.3 Interacción de los usuarios con la información.

1.5 Definición del proyecto

La implementación de un *cluster* de alta disponibilidad de bases de datos para la geDGAPA considera tres etapas subsecuentes de desarrollo, las cuales son:

1. Implementación
2. Migración
3. Pruebas y monitoreo

Cada una de las etapas mencionadas anteriormente tiene su importancia y son parte medular para la implementación y funcionamiento del *cluster*. A continuación se describirá cada una de ellas, para su desarrollo posterior en el capítulo 3.

Implementación

La implementación es la primera etapa a realizar y **consiste** en montar tres servidores Sun Fire X4200 con sistema operativo Ubuntu Server 8.

Posteriormente se implementará la seguridad en los tres servidores, mediante la configuración del *firewall* por medio de *iptables*, donde los puertos a ocupar se mantendrán abiertos y todos los demás se les denegará el acceso.

Después se procederá a instalar el *software* de licencia libre PostgreSQL como **DBMS** (Sistema Manejador de Bases de Datos) a utilizar y posterior a la migración se instalará el *software* que proporcionará la estructura de replicación del *cluster*, Slony versión 1.2.15 y por último el *software* Pgpool II que se encargará del proceso de *failover* cuando se requiera hacer un cambio de nodo maestro.

Migración

La migración es un proceso en el cual hay que tener especial atención, debido a la importancia de los datos que se manejan en dicho proceso.

Actualmente para la geDGAPA el DBMS utilizado es PostgreSQL versión 7.4.6 y el *cluster* requiere tener una versión más actual, por lo tanto se instalará la versión 8.2.4.

Para garantizar que el proceso de migración se realice de manera correcta, se utilizará el software DBComparer y DataComparer, que fungirán como la herramienta que nos permitirá verificar que el número de registros de la base origen sea el mismo en estructura y contenido en la base destino.

Pruebas y Monitoreo

Inmediatamente después de la implementación y migración es necesario realizar pruebas para observar si el *cluster* está funcionando de la manera deseada, para ello se necesitará emplear un software de monitoreo, en nuestro caso, se utilizará Mon, Power Alert y Smart Mon Tools, herramientas de monitoreo de licencia libre, para poder verificar si de verdad el sistema está trabajando como se había esperado.

Al concluir las etapas de pruebas y monitoreo, el sistema se pondrá al servicio de los usuarios de la gestión electrónica de la DGAPA.

Con la implementación de un *cluster* de bases de datos sobre PostgreSQL montado en tres servidores Sun Fire X4200, se espera obtener un sistema de alta disponibilidad para la geDGAPA en el cual cada uno de los programas manejados por la dependencia tenga un tiempo de respuesta óptimo. Por lo anterior, implementar el *cluster* beneficiará a los programas manejados por la geDGAPA y a toda la comunidad universitaria que tiene acceso a ellos.

Capítulo 2

Marco Teórico

Capítulo 2 Marco Teórico

2.1 Introducción

Es necesario conocer de manera general lo que es un *cluster*, su funcionamiento y componentes para poder entender el desarrollo de esta tesis; por lo tanto, este capítulo se encargará de proporcionar al lector un panorama teórico de los aspectos fundamentales para la implementación de un *cluster*, con la finalidad de establecer bases claras y firmes para el entendimiento de los capítulos posteriores.

2.2 Requerimientos de un sistema

Los sistemas de cómputo son la central estratégica de negocio de las organizaciones y tal vez la representación más visible de la misma. Tal es el caso en las áreas de ventas, finanzas, industria, educación, gobierno entre muchas otras. Mientras la tecnología avanza se vuelve más disponible y el rol que desempeñan se vuelve más vital.

Las necesidades de los sistemas de información que se han presentado en los últimos años, provocaron un acelerado uso de nuevas tecnologías para obtener sistemas más poderosos que puedan soportar el trabajo pesado, como lo es la administración de una base de datos y un servidor de correo. Estos son ejemplos de algunos servicios que buscan obtener un alto grado de eficiencia y disponibilidad, que muchas veces resulta difícil de obtener sin tomar en cuenta que siempre se quiere el mejor resultado al menor costo posible.

La información almacenada en estos sistemas, se ha convertido en un elemento de vital de importancia para todos aquellos organismos que manejen grandes volúmenes de datos. Para ello, la tecnología de la información, se enfoca en resolver problemas de eficiencia como:

- El proporcionar facilidad de acceso a la información, satisfaciendo necesidades de los usuarios sin importar donde se encuentren dentro de la red de la organización.
- Proporcionar facilidad en la administración de los sistemas, de manera flexible, adaptable a cambios que se vayan presentado, sin que represente un costo elevado y siga siendo eficaz.
- Proporcionar gran capacidad de almacenamiento.

En consecuencia, el *cluster* es una alternativa de solución dentro de los sistemas de alto rendimiento de hoy en día.

Actualmente un sistema se evalúa por las habilidades más eficaces que posea, tales como la disponibilidad, confiabilidad, entre otras, que son necesarias en las operaciones diarias para proporcionar la información requerida por los usuarios.

A continuación se presenta un análisis de los requerimientos que debe presentar un sistema de hoy en día. En la figura 2.1 se muestran los requerimientos que se desean idealmente en los sistemas modernos en una empresa. [12]

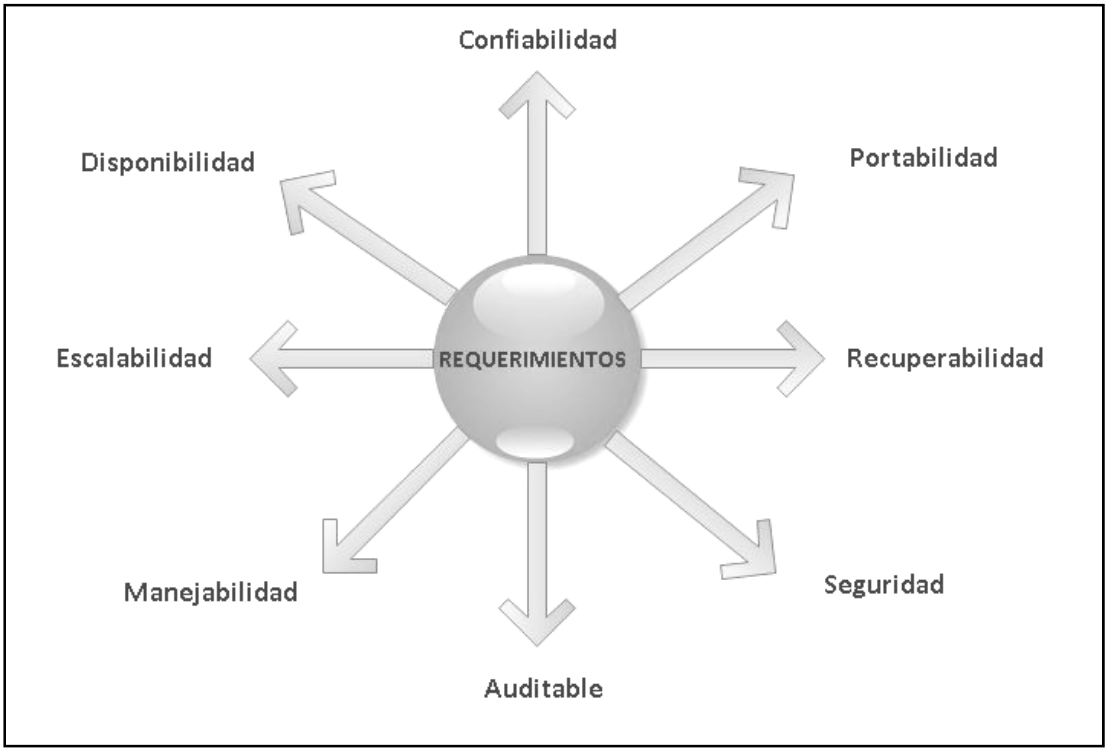


Figura 2.1 Principales requerimientos de un sistema

Es importante tener un concepto más específico de cada uno de estos requerimientos, por ello se describirán a continuación:

Confiabilidad

Se dice que un sistema es confiable cuando por ejemplo un usuario se conecta a un sistema para hacer un proceso específico, el sistema debe garantizar una respuesta. Estos requerimientos aplican a ambos lados de un sistema, el de la aplicación en sí y a la lógica de la aplicación específica para el negocio.

Portabilidad

La portabilidad es un requerimiento de gran importancia y está relacionado actualmente con los sistemas modernos. La portabilidad debe asegurar que el sistema será adaptable a distintas plataformas según las necesidades de crecimiento del negocio. Por lo tanto la infraestructura de un sistema debe ser organizada de tal manera que los cambios a realizar sean los mínimos sobre la plataforma del *hardware*.

Recuperabilidad

La recuperabilidad se refiere a que un sistema debe ser recuperable de fallas con un tiempo fuera de servicio mínimo. La recuperabilidad en bases de datos está directamente

[12] Vallath,Murali,2004,Oracle real application clusters.

relacionada con la estrategia de respaldo del sistema. Una buena estrategia de respaldo se mide en el tiempo de recuperación del sistema. Idealmente el sistema debe estar arriba y corriendo rápidamente después de haber ocurrido una falla.

Seguridad

Los días en que las aplicaciones eran usadas en un comunidad pequeña de usuarios se han quedado atrás, en la arquitectura cliente/servidor, las aplicaciones eran usadas por un pequeño número de usuarios y estos usuarios eran fácilmente identificables y pertenecían a la misma organización.

Actualmente los sistemas basados en internet tienen bases de datos accesibles para todo el mundo. En consecuencia, la seguridad de los datos se ha convertido extremadamente importante. Los datos en una organización son vitales y deben estar adecuadamente protegidos de hackers.

Auditable

La auditoria de los datos se refiere a la capacidad de regresar la información suficiente con respecto a la creación de los datos, es decir, conocer quien creó dichos datos, por que se crearon , quien los modificó, cuando fueron modificados , etc. La auditabilidad es un requerimiento importante y su existencia se remonta desde que las computadoras fueron puestas en operación de manera comercial.

Manejabilidad

La manejabilidad abarca varias áreas en diferentes aspectos. Los sistemas que se desarrollan deben además ser de fácil mantenimiento y manejabilidad por cada área del sistema. La manejabilidad se refiere al monitoreo, afinamiento y organización del sistema.

Escalabilidad

La escalabilidad es típicamente definida como la habilidad de un sistema de adaptarse a las necesidades de crecimiento del negocio.

Disponibilidad

La disponibilidad es otro importante requerimiento de los sistemas basados en internet de hoy en día, la disponibilidad va fuertemente interrelacionada con la confiabilidad, cuando la disponibilidad es baja se convierte en una pérdida muy costosa para la empresa cuando el sistema se encuentra fuera de servicio, en cambio, cuando la disponibilidad es alta, el servicio se mantendrá disponible y por lo tanto dará confiabilidad a los clientes.

La disponibilidad es medida por la cantidad de tiempo en el que el sistema está disponible para realizar operaciones y por **sistema** no solamente aplica a la capa de datos o la capa de aplicación del sistema, sino al sistema total de la empresa, esto implica que cada componente del equipo, incluyendo redes, servidores, etc., deben ser considerados para tener disponibilidad. Hacer todas las capas del sistema disponible significa que cada una de las capas debe proveer redundancia de *hardware*, ayudando a proveer servicio continuo cuando uno de los componentes llegue a fallar.

En conclusión los negocios modernos han empezado a requerir factores mas tangibles en un sistema, debido al *boom* de la internet, el nivel de requerimientos como la disponibilidad, escalabilidad, confiabilidad, entre otros ya mencionados anteriormente, juegan un rol importante en donde los usuarios dependen directamente de un sistema de cómputo.

2.3 Cluster

El concepto de *cluster* nació cuando los pioneros de la supercomputación intentaban difundir diferentes procesos entre varias computadoras, para luego poder recoger los resultados que dichos procesos debían producir.

Así pues, podríamos definir a un *cluster* como un conjunto de pc's o estaciones de trabajo (llamados nodos) que interconectadas con dispositivos de alta velocidad por alguna tecnología de red, trabajan como un solo recurso integrado de cómputo y aparecen ante clientes y aplicaciones como un solo sistema, con el objetivo de satisfacer alguna necesidad de: alta disponibilidad, balanceo de carga, alto rendimiento o alguna combinación de las anteriores.

Para que un *cluster* funcione como tal, no basta con sólo conectar entre sí las computadoras, sino que es necesario proveer un sistema de administración del *cluster,* el cual se encargue de interactuar con el usuario y los procesos que corren en él para optimizar su funcionamiento.

El ***clustering*** es una arquitectura que mantiene un sistema corriendo en el caso de que ocurra una falla en el sistema. El *cluster* brinda máxima escalabilidad por medio de un arreglo de servidores que funcionan como un solo recurso de computación. El *cluster* tiene el potencial de establecer una relación costo/desempeño por arriba de un *mainframe*, en disponibilidad, escalabilidad, manejabilidad y recuperabilidad. El *cluster* juega un papel esencial en sistemas que requieren soluciones de alta disponibilidad ininterrumpida.

Al utilizar la tecnología de *clustering*, los usuarios podrán agregar gradualmente a sus sistemas elementos que requieran para aumentar su productividad y de esta manera satisfacer los requerimientos de procesamiento y almacenamiento.

2.3.1 Orígenes y un poco de historia

La idea de enlazar varias computadoras con un propósito común no es nueva, en las décadas de los 50's y 60's la Fuerza Aérea de los Estados Unidos estableció la red de defensa aérea más grande hasta ese momento: el sistema SAGE (*Semi-Automatic Ground Enviroment*). Este sistema coordinaba varias estaciones de radar con el objetivo de detectar amenazas aéreas (aviones bombarderos soviéticos, misiles intercontinentales etc.) y guiar misiles para su intercepción y destrucción. SAGE estuvo en funcionamiento durante 20 años (1963-1983) bajo las órdenes del Comando de Defensa Aeroespacial de Norte América (NORAD) y la sala de operaciones del sistema SAGE se muestra en la figura 2.2.[13]

[13] Gómez Macías, 2008, Diseño de un cluster Beowulf para realizar cálculos Monte Carlo.

Figura 2.2 Sala de operaciones del sistema de defensa SAGE.

Fue el año de 1984 cuando la compañía DEC utilizó por primera vez el término *cluster* para representar un conjunto coordinado de computadoras independientes enfocadas a una tarea específica y que lograrían la atención del mercado con el lanzamiento de sus VAXclusters, sistemas conformados por la conexión de varias minicomputadoras VAX funcionando en conjunto, tal como se puede apreciar en la figura 2.3.

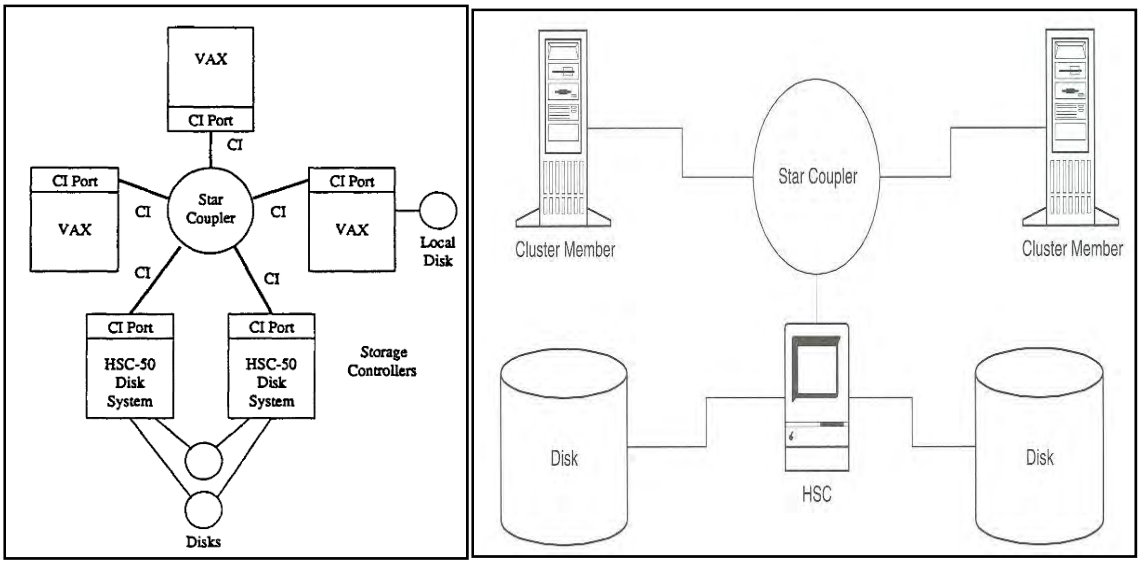


Fig. 2.3 Configuración típica del VAXCluster.[14]

[14] Vallath,Murali,2004,Oracle real application clusters.

El bus principal del *cluster* es un dispositivo llamado como *star coupler*, el acoplador estrella acepta conexiones de los procesadores y de las unidades especiales de control, como se aprecia en el diagrama no hay redundancia en el acoplador estrella, en si es un dispositivo pasivo con una fuente de poder. El acoplador estrella puede ser un panel de parcheo o un *hub*.

Las unidades especiales de control son llamados como **HSCs** en su traducción Almacenamiento Jerárquico de Controladores (*Hierarchical Storage Controllers*). EL HSC maneja el disco y los dispositivos de entrada-salida.

En función de proveer redundancia de disco, el VAXCluster, mantiene los datos en los dos discos, cualquier escritura de datos es realizada en ambos discos, pero las operaciones de lectura son consultadas en un solo disco, ya que los dos contienen la misma información. Como todos los procesadores en el VAXCluster tienen acceso al HSC a través del acoplador estrella, no hay ningún punto de falla en el *hardware*.

Pero el desarrollo más significativo y recordado para la tecnología de *clusters* se llevaría a cabo en 1993 cuando el Centro de vuelos espaciales Godarth de la NASA trabajó en el proyecto "Ciencias de la Tierra y el Espacio" a cargo de Jim Fisher. El centro de cómputo del proyecto contaba con computadoras paralelas llamadas Cray, MasPar y Convex, con suficiente capacidad computacional para realizar simulaciones en un tiempo aceptable para aquella época. El principal problema era que los resultados obtenidos eran demasiado grandes para poder analizarlos desde una estación de trabajo científica. Debido a esto, las supercomputadoras no sólo se utilizaban para generar los resultados sino también para organizarlos y analizarlos, propiciando un desperdicio en posibles tiempos de simulación.

Fue entonces cuando Jim Fisher comprendió que se necesitaba una especie de supercomputadora personal y decidió que su diseño y creación sería otra meta de su proyecto. En esa época el Dr. Thomas Sterling trabajaba en el proyecto a cargo de Jim Fisher y fue a él a quién se le ocurrió la idea de apilar una cierta cantidad de computadoras de bajo costo para así obtener mayores capacidades de cómputo. Después de plantear su propuesta ésta fue aceptada y recibió los fondos necesarios para empezar a trabajar.

Al final el proyecto fue un éxito; con sólo $40,000 dólares se instaló y configuró el *cluster* llamado Wiglaf conformado por 16 computadoras 80486 a 100MHz cada una con 32 MB en RAM y un Gigabyte en disco duro. Ejecutando el sistema operativo GNU LINUX, todas conectadas por medio de una red Ethernet 10-Base-T con unión de canales, se alcanzaron velocidades de hasta 70 Mflops. La era del supercómputo de bajo costo había comenzado y desde entonces cientos de universidades han recurrido a los *clusters* como una útil herramienta de investigación.

2.3.2 Funcionamiento de un *cluster*

El *cluster* trabaja primeramente a través de un mecanismo llamado ***polling.*** *Polling* es una actividad similar al ***ping,*** donde un mensaje es enviado a un dispositivo y donde la respuesta es examinada. Si una respuesta es exitosa es recibida entonces el *polling* es exitoso, de lo contrario se determina que ocurre un problema. Todos los servidores participan en la configuración del *cluster*, mantienen un *polling* entre cada uno de ellos para determinar si algún servidor no funciona

correctamente. Si algo no está correcto, el proceso de ***failover*** entra en acción. El *polling* confía en las conexiones de red, es decir la LAN (Red de Área Local).[15]

La figura 2.4 ilustra la configuración de un *cluster* con dos nodos, con la interconexión del *cluster*, el bus de conexión SCSI para la LAN, la primera es usada para realizar el *polling* y determinar que nodos están disponibles mientras que la segunda es usada por el sistema para determinar, desde el chequeo inicial con la interconexión del *cluster*, cuáles de los nodos participantes en el *cluster* están disponibles. Una vez que se detecta una señal de falla, otra validación es ejecutada usando la conexión de red LAN antes de proceder a realizar la operación de *failover*.

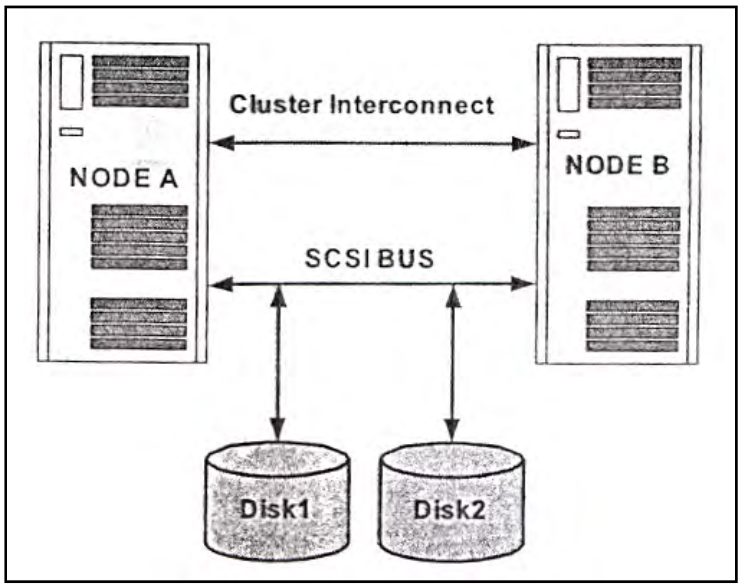


Figura 2.4 Configuración del cluster con dos nodos.[16]

Un *polling* de conexión es mandado al otro nodo en el *cluster* para ver si el nodo está disponible, si el nodo, no responde con un tiempo especificado internamente (tiempo determinado mediante el valor del parámetro del ***heartbeat***), el *polling* trata de encontrar al nodo dentro de la LAN. El parámetro de tiempo fuera del *hearbeat* define un periodo de tiempo fuera después de que no hay respuesta alguna del nodo, se asume que el nodo no está disponible. La falla o el éxito en el *polling*, indica si el problema se encuentra en el nodo ó en la interconexión del dispositivo.

Si el nodo al que se le mando el *polling* determina a través de la interconexión del *cluster* y de la LAN, que el otro nodo esta fuera de servicio, procederá a tomar el control de los recursos del disco que dicho nodo tenía, para mantenerlos disponibles en el sistema. Cuando el nodo fuera de servicio se recupere y vuelva a estar disponible, el nodo sustituto abandona los recursos del disco a su legítimo dueño, tan pronto como no haya nodos tratando de acceder a él. Este es un proceso automático y conocido como *failover.*

[15] Vallath, Murali, 2004, Oracle real application clusters.
[16] Idem.

Para que el proceso de *failover* entre en acción se puede deber a las siguientes causas:

- Que la conexión del *polling* falle.
- Que la LAN falle.
- Que el nodo que realizó el *polling* tenga éxito y tome el control de los recursos del nodo faltante del *cluster*.

2.3.3 *Downtime*

El costo que representa una falla en el sistema puede ser elevado ya que degrada el rendimiento, funcionalidad y disponibilidad. Muchos eventos causan impacto negativo en el momento en que se están guardando las aplicaciones que se encuentran trabajando en ese instante. En esta parte, se tratará de conocer cuáles son las causas más comunes por las cuales el sistema falla. Estas causas se agrupan en cuatro conjuntos principales:[17]

1. Problemas de *hardware*. Las principales causas de una caída del sistema provocadas por problemas de *hardware* se atribuyen a:

- Fallas en los discos
- Variaciones de voltaje
- Fallas en los ventiladores provocando calentamiento de los equipos
- Fallas en la memoria y en los buses de transmisión de la información
- Errores en las controladoras de arreglos de discos o en los adaptadores de red.

Las fallas en los discos y las variaciones de voltaje, se consideran como las más frecuentes causas de fallas.

2. Problemas de *software*. Las fallas en el *software* son las responsables de la mayor parte de las caídas del sistema. Entre los problemas de *software* más comunes están los siguientes:

- Fallas del sistema operativo
- Fallas en aplicaciones
- Defectos de software

Cuando una aplicación falla y no responde, se considera como el problema más común y puede llegar a provocar otros problemas con el sistema operativo.

3. Mantenimiento y actualizaciones. Las caídas del sistema provocadas por el mantenimiento o por realizar actualización, se encuentran en el segundo lugar de fallas más comunes. Pero en este caso, el administrador del sistema dará por finalizado el servicio por las siguientes razones:

- Mantenimiento de *hardware*
- Reparación o reemplazo de *hardware*
- Mantenimiento de *software*

[17] Hernández Rodríguez, 2003, Implementación de un sistema de alta disponibilidad basado en tecnología de cluster.

- Respaldos

4. Eventos naturales. Solamente cuando se presenta alguna de las siguientes causas ambientales consideradas como severas, se provocaría la caída del sistema. Algunos de los problemas ambientales que se presentan son:

- Humedad excesiva
- Desastres naturales (terremotos, huracanes, tornados)
- Apagones

La tabla 2.1 y la figura 2.5 muestran un análisis de la disponibilidad y el tiempo permitido de horas fuera de servicio (*downtime*).[18]

Tabla 2.1 Disponibilidad (%) vs *downtime*

Disponibilidad	Tiempo fuera de servicio por año (*downtime*)
99.995%	0.5 horas
99.97%	2.5 horas
99.8%	17.5 horas
99.5%	43.2horas ó 1.8 días
99%	88.8 horas ó 3.7 días
98%	175.2 horas ó 7.3 días
96%	350.4 horas ó 14.6 días

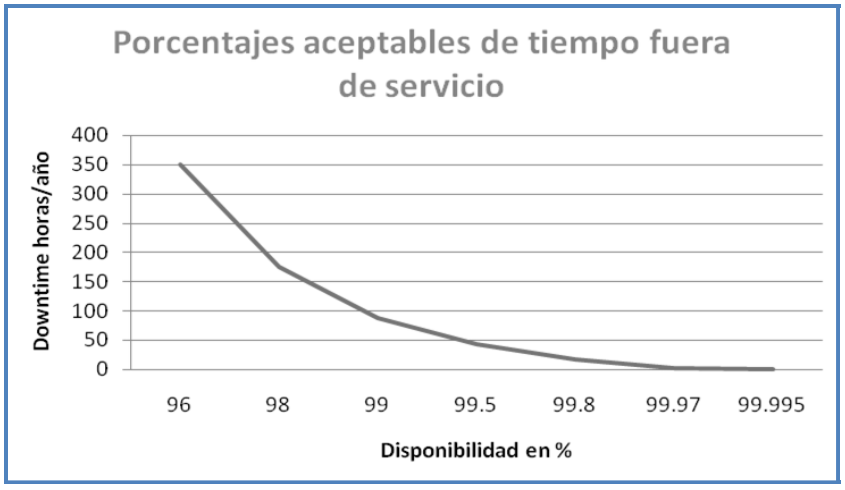


Figura 2.5 Gráfica comparativa de disponibilidad vs *downtime*.

[18] Vallath,Murali,2004, Oracle real application clusters.

2.3.4 Beneficios de un *cluster*

A continuación se mencionará las principales ventajas que representa tener un *cluster* dentro de los procesos de una organización.[19]

Alta disponibilidad

La disponibilidad es la medida o el factor con el cual un sistema de cómputo puede ser evaluado para decidir si sigue funcionando y prestando servicios a los clientes. Ya que si uno de los nodos del *cluster* falla o si el administrador piensa darlo de baja, el otro nodo debe tomar el control de los servicios para que sigan trabajando y no detener las operaciones. La disponibilidad depende de que el sistema sea apto para prevenir y recuperarse de las fallas que se presenten. Un sistema de alta disponibilidad incluye aspectos de tolerancia a fallas y para lo cual todos los componentes del equipo o del sistema deben contar con dispositivos redundantes.

Escalabilidad

Añadiendo otra característica que nos puede dar un sistema de alta disponibilidad, los *clusters* también proporcionan escalabilidad. Y se le considera como la capacidad de expandir los recursos en cada etapa, incrementado el acceso a los recursos existentes y aumentando la disponibilidad de éstos. La escalabilidad facilita la inversión de *software y de hardware* en el sistema, ya que al estar actualizado asegura la protección de los recursos. El problema que surgía con este tipo de soluciones es que con el paso del tiempo aparecían nuevos equipos con tecnología más avanzada y para poder mantener el sistema al día, se tenía que reemplazar el equipo con uno nuevo.

Balanceo de cargas

En cualquier servidor, las aplicaciones y los recursos pueden fallar debido a una excesiva carga de trabajo, ya que sobrepasaron los recursos del sistema, pero en el *cluster* se balancea esta carga mediante un proceso interno de distribución entre los recursos disponibles. En el *cluster* todas las tareas son distribuidas entre nodos, evitando el exceso de carga en un solo nodo y previniendo fallas por este tipo de problemas. Y en caso de que uno de los nodos falle, la carga total de trabajo se pasará automáticamente al otro nodo. El balanceo de cargas es automático y también es controlado por el administrador del sistema y no afecta de ninguna manera el desempeño del sistema ni el trabajo que estuvieran realizando los usuarios en ese momento.

Un *cluster* puede presentarse como una solución de especial interés sobre todo a nivel de empresas, las cuales pueden aprovecharse de estas especiales características de computación para mantener sus equipos actualizados por un precio bastante más económico que el que les supondría actualizar todos sus equipos informáticos y con unas capacidades de computación que en muchos casos pueden llegar a superar a *hardware* de última generación.

[19] Hernández Rodríguez, 2003, Implementación de un sistema de alta disponibilidad basado en tecnología de cluster.

Administración sencilla

La administración de los equipos se realiza por medio de herramientas administrativas, con la finalidad de monitorear los recursos y las actividades del servidor, estas tareas normalmente son realizadas por el administrador.

Los *clusters* ofrecen las siguientes ventajas a un costo relativamente bajo, en la tabla 2.2 se observa cada una de ellas con una breve descripción:

Tabla 2.2 Resumen de las ventajas del cluster

Ventaja	Descripción
Alta Disponibilidad (*High Availability*)	Es la capacidad de proveer disponibilidad y confiabilidad en el momento que se necesite, esta puede ser provista mediante la detección de la falla de un nodo o servicio y reconfigurando el sistema apropiadamente para que la carga de trabajo pueda ser manejada por los restantes nodos del *cluster*.
Escalabilidad	Es la capacidad de modificar el sistema de manera conveniente a las crecientes necesidades que puedan surgir en un período de tiempo, es decir, que es un sistema que va a estar vigente en un futuro.
Balanceo de cargas	Es la capacidad de repartir la carga de trabajo entre los nodos del *cluster*.
Administración sencilla	El *cluster* tiene la ventaja de poder utilizar distintas herramientas administrativas que faciliten el manejo de los procesos que corren dentro de él.
Alto Rendimiento (*High Performance*)	Es la capacidad que se tiene para ejecutar tareas que requieren de gran capacidad de cómputo.
Fiabilidad	Es la capacidad de proveer el funcionamiento correcto.
Alta Eficiencia (*High Throughput*)	Es la capacidad que se tiene para ejecutar la mayor cantidad de tareas en el menor tiempo posible.
Estabilidad	Es la capacidad de un equipo de hacer frente a volúmenes de trabajo cada vez mayores, sin dejar por ello de prestar un nivel de rendimiento aceptable.

2.3.5 Tipos de clusters

Clasificación de los *clusters* según su uso

Básicamente podríamos distinguir tres tipos de *clusters* atendiendo al uso que queramos darle:[20]

[20] Gómez Macías, 2008, Diseño de un cluster Beowulf para realizar cálculos Monte Carlo.

1. *Clusters* de balanceo de carga

Son *cluste*rs que distribuyen su carga de trabajo a través de múltiples nodos. Todos los nodos son capaces de responder a peticiones externas debido a que ejecutan los mismos programas y en caso de que se presente la falla de un nodo las peticiones son distribuidas entre los nodos activos restantes. Este tipo de *clusters* tiene la ventaja de que los nodos pueden estar instalados en distintos puntos geográficos además de que su instalación es relativamente fácil. La tecnología web se ha visto altamente beneficiada con la utilización de *clusters* de balanceo de carga para satisfacer a todas las peticiones en sus servidores.

En la figura 2.6 se muestra la estructura de un *cluster* de balanceo de carga.

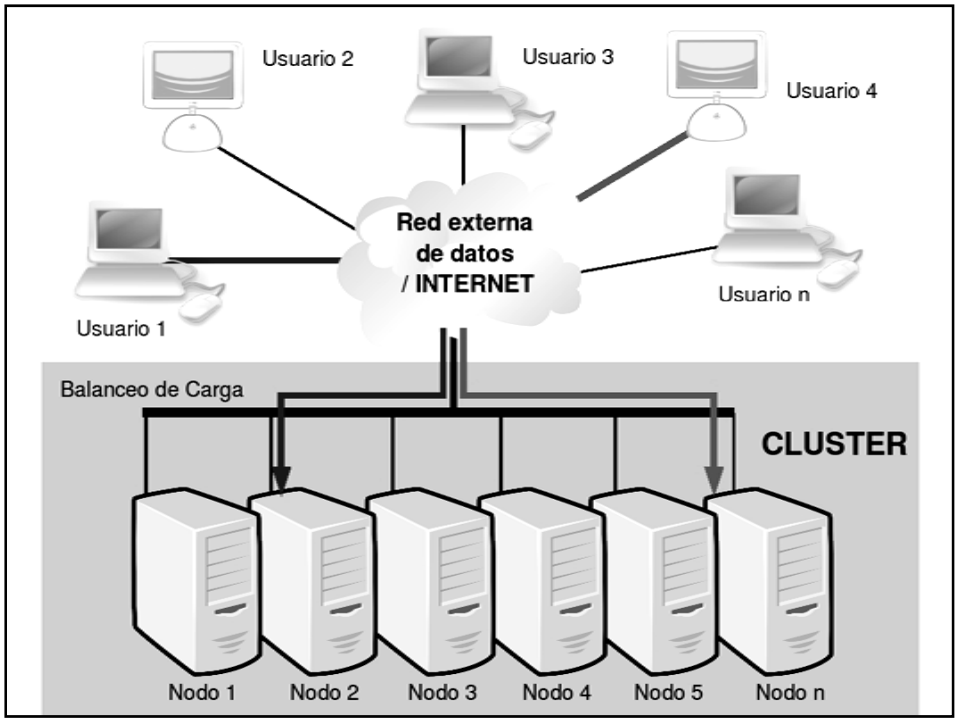


Figura 2.6 *Cluster* de balanceo de carga.[21]

2. *Clusters* de alta disponibilidad

Son *clusters* utilizados en aplicaciones de misión crítica; están compuestos por varios nodos donde un grupo de nodos principal se encarga de brindar algún servicio mientras que un segundo grupo de nodos monitorea el correcto funcionamiento del grupo de nodos principal. Así cuando el grupo principal falle, el segundo grupo entrará como un reemplazo inmediato evitando así la suspensión del servicio. Su utilización en conjunto con los *clusters* de balanceo de carga es bastante aplicada. En sí, consiste en la conexión de una o varias computadoras conectadas en red utilizándose una conexión ***heartbeat*** para monitorear cual de sus servicios está en uso, así como la sustitución de una máquina por otra cuando uno de sus servicios haya caído.

[21] Idem.

En la figura 2.7 se muestra la estructura de un *cluster* de alta disponibilidad.

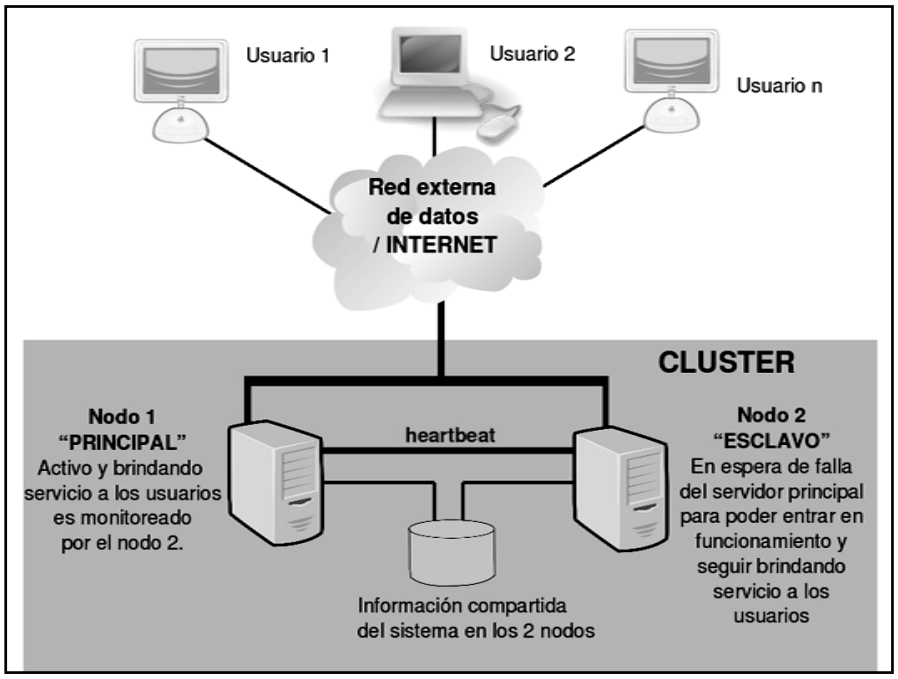


Figura 2.7 Cluster de alta disponibilidad.[22]

3. *Clusters* de alto rendimiento

Son *clusters* utilizados para resolver problemas que requieren de grandes capacidades de cálculo. Están compuestos por varios nodos que ejecutan código de manera paralela con el objetivo de obtener resultados en un menor tiempo. Es el tipo de *cluster* más popular y utilizado desde la aparición de los *clusters* tipo **Beowulf**. En sí, son *clusters* destinados al alto rendimiento, tienen capacidad muy alta de proceso para cómputo de grandes volúmenes de datos.

En la figura 2.8 se muestra la estructura de un *cluster* de alto rendimiento.

[22] Gómez Macías, 2008, Diseño de un cluster Beowulf para realizar cálculos Monte Carlo.

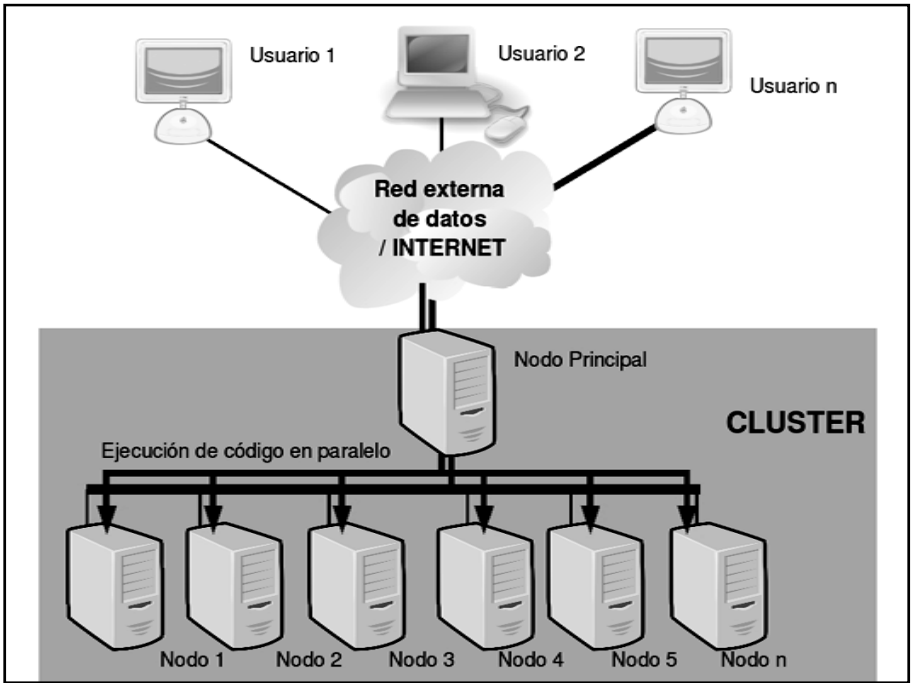


Figura 2.8 Cluster de alto rendimiento.[23]

Clasificación de los *clusters* según sus componentes

Existen distintos tipos de *clusters* según sus componentes, se clasifican en tres tipos que se mencionan a continuación en la tabla 2.3:

Tabla 2.3 Clasificación de los clusters según sus componentes

Clasificación	Descripción
Homogéneo	Se vale de la misma configuración de *hardware* y sistema operativo en todos sus ordenadores.
Semi-Homogéneo	Posee un rendimiento disímil pero la arquitectura general es similar.
Heterogéneo	Tiene distinto *hardware* y sistema operativo, esto los hace más económicos y rendidores.

2.3.6 Componentes de un Cluster

En general, un *cluster* necesita de varios componentes de *software* y *hardware* para poder funcionar, como se aprecia en la figura 2.9:

[23] Gómez Macías, 2008, Diseño de un cluster Beowulf para realizar cálculos Monte Carlo.

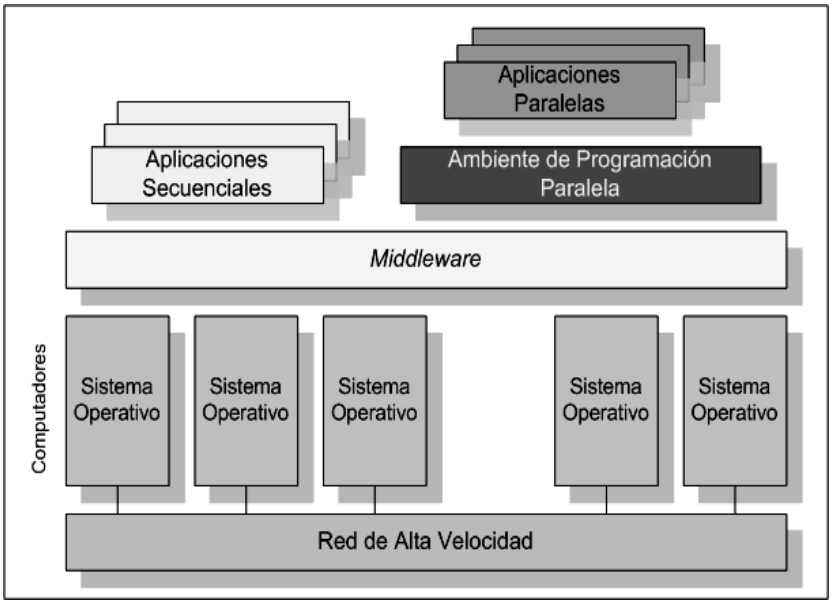


Figura 2.9 Componentes de un cluster.[24]

A continuación se describe de manera breve cada componente del *cluster*.

Nodos

Pueden ser simples computadoras, sistemas multi procesador o estaciones de trabajo.

Sistemas Operativos

Debe ser de uso amigable, de fácil acceso y permitir múltiples procesos y usuarios.

Conexiones de red

Los nodos de un *cluster* pueden conectarse mediante una simple red Ethernet o puede utilizar tecnologías especiales de alta velocidad como *Fast Ethernet* ó *Gigabit Ethernet*.

Middleware

El *middlewar*e es un software que generalmente actúa entre el sistema operativo y las aplicaciones con la finalidad de proveer:

- Un interfaz único de acceso al sistema, denominado SSI (*Single System Image*), el cual genera la sensación al usuario de que utiliza una única computadora muy potente.

[24] http://clusterfie.epn.edu.ec/clusters/Definiciones/definiciones.html
Componentes de un cluster.

- Herramientas para la optimización y mantenimiento del sistema: migración de procesos, *checkpoint-restart* (detener uno o varios procesos, migrarlos a otro nodo y continuar su funcionamiento), balanceo de carga y tolerancia a fallos.

- Escalabilidad: debe poder detectar automáticamente nuevos nodos conectados al *cluster* para proceder a su utilización.

2.3.7 Arquitectura de un cluster

La arquitectura de un *cluster* está integrada por cinco capas, como se muestra en la figura 2.10:

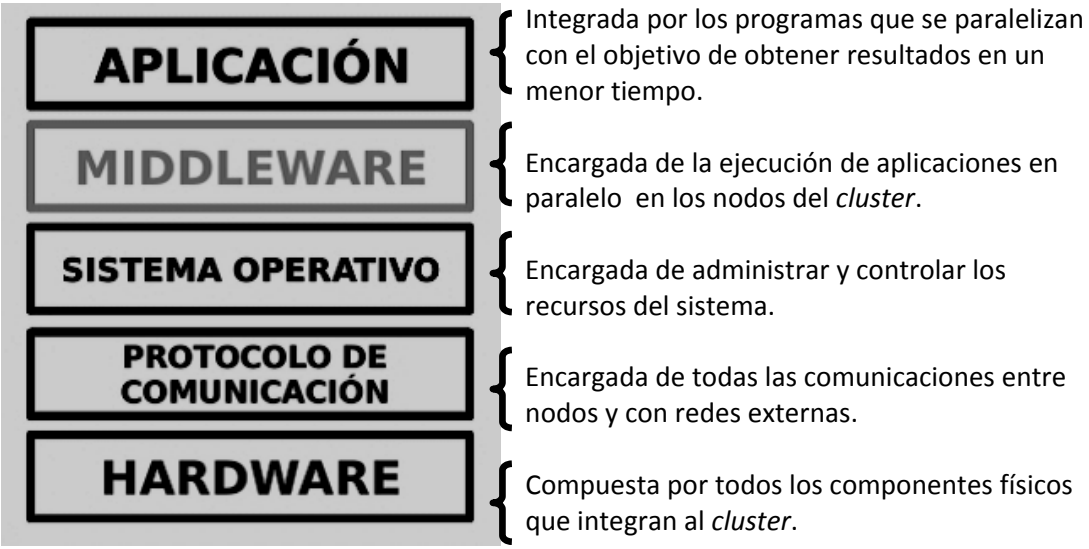


Fig.2.10 Arquitectura de un cluster

La elección de los elementos que integran cada una de las capas definirá el desempeño final que tendrá un *cluster* al ejecutar aplicaciones. Antes de poder diseñar un *cluster* de alto rendimiento es necesario desarrollar y entender cada una de las capas tal como se presenta a continuación:

Capa de *hardware*

El *hardware* de un *cluster* de cómputo es el conjunto de componentes físicos que integran a esta tecnología. Una buena elección de *hardware* es crucial para el desempeño final que tendrá el *cluster*. A continuación se presenta una breve descripción del principal *hardware* que integra los nodos de un *cluster* de alto rendimiento.

Microprocesador

El microprocesador es el componente encargado de interpretar datos y ejecutar las instrucciones de los programas, también coordina a los demás componentes de una computadora. El microprocesador es comúnmente llamado el cerebro de la computadora pero dada la

incapacidad de este para *pensar* una descripción más acertada sería decir que el microprocesador es una calculadora extremadamente veloz (miles de millones de operaciones por segundo) con la capacidad de realizar operaciones aritméticas simples y almacenar sus resultados. Las características que se deben tomar en cuenta para su elección se presentan a continuación:

Memoria de acceso aleatorio

Conocida simplemente como memoria **RAM** (*Random Access Memory*) es la memoria que almacena instrucciones y datos de manera temporal (es memoria volátil) para realizar tareas dentro de una computadora. El objetivo de su utilización es la reducción del tiempo de ejecución de los programas ya que el microprocesador tarda un menor tiempo en extraer datos de la memoria RAM que del disco duro. La memoria RAM se divide en dos tipos principales:

- **Memoria de Acceso Aleatorio Estática (SRAM):** es una memoria fabricada a partir de componentes que no requieren de refresco, logrando así que la memoria pueda trabajar casi a la misma velocidad del microprocesador. Debido a su costo, consumo de energía, tamaño de componentes y calor generado, supera por mucho a los de la DRAM.

- **Memoria de Acceso Aleatorio Dinámica (DRAM*):*** es una memoria fabricada a partir de transistores acoplados con condensadores, siendo éstos los que dependiendo de su estado (cargado o descargado) representan los unos y ceros a almacenar. Es una memoria mucho más lenta a comparación de la SRAM pero sus bajos costos en fabricación la han convertido en la memoria más comercial y utilizada.

Dentro de las características que hay que tomar en cuenta para la elección de la memoria RAM en la construcción de un *cluster* se tienen las siguientes:

1. La velocidad a la que trabaja la memoria debe estar soportada por la placa base o *motherboard*.
2. En base a la aplicación que tendrá el *cluster* se deberá elegir la cantidad de memoria RAM en el nodo.

Dispositivos de almacenamiento permanente

El disco duro es el principal dispositivo de almacenamiento permanente dentro del nodo, está compuesto por una serie de platos que giran a gran velocidad (7200 rpms hasta 15,000 rpms) dentro de una carcasa. Cada plato cuenta con un cabezal que se desplaza a lo largo de toda su superficie con el objetivo de leer o escribir impulsos magnéticos. La elección del disco duro para un nodo de *cluster* depende de 3 aspectos principales:

- El tipo de conexión con el que cuenta la placa base del nodo (**SATA, IDE, SCSI**).
- La capacidad de almacenamiento requerida en el nodo (desde unos cuantos **Gigabits** hasta cientos).
- Estudiar la posibilidad de no comprar discos duros (a excepción del que necesita el nodo principal) y realizar un diseño de *cluster* donde todo el sistema sea exportado del nodo principal.

Periféricos

Son el conjunto de dispositivos que realizan operaciones de entrada/salida complementarias al procesamiento de datos que realiza el microprocesador de un nodo. Los periféricos se dividen de la siguiente manera:

- **Dispositivos de entrada:** ingresan datos a la computadora para que sean procesados por el cpu (teclado, mouse, escáner, micrófono, etc.)
- **Dispositivos de salida:** reciben datos procesados por el cpu y se encargan de desplegarlos de manera comprensible al usuario (monitor, impresora, etc.)
- **Dispositivos de almacenamiento:** almacenan datos de manera permanente (disco duro, memoria flash, lectores y grabadores de CD, DVD, HD-DVD, etc.)
- **Dispositivos de comunicación:** permiten la comunicación entre computadoras (tarjeta de red, fax-módem, etc.)

Cabe señalar que los nodos requieren de muy pocos periféricos para ser funcionales, en principio con la tarjeta de red, la placa base y el disco duro sería suficiente.

Capa de Protocolo de comunicación

Modelo de referencia TCP/IP

El Protocolo de Internet (**IP**) y el Protocolo de Transmisión (**TCP**), fueron desarrollados inicialmente en 1973 por el informático estadounidense Vinton Cerf como parte de un proyecto dirigido por el ingeniero norteamericano Robert Kahn y patrocinado por la Agencia de Programas Avanzados de Investigación (ARPA, siglas en inglés) del Departamento Estadounidense de Defensa.[25]

Tiempo después este conjunto de protocolos serían la base de Internet permitiendo que todo tipo de redes, con *hardware* y software variado, pudieran conectarse. Se le conoce como conjunto de protocolos TCP/IP debido a que sus dos protocolos más importantes son el IP y el TCP pero en realidad el conjunto está integrado por más de 100 protocolos. Una de las grandes ventajas del TCP/IP, es que es compatible con cualquier sistema operativo y con cualquier tipo de *hardware*.[26]

La arquitectura del modelo TCP/IP se agrupa en cuatro capas como se aprecia en la figura 2.11:

[25] http://www.slideshare.net/chel0nline/tcpip-presentation
TCP/IP vs Modelo OSI.

[26] http://usuarios.lycos.es/janjo/janjo1.html
Protocolo TCP/IP por Miguel Alenjandro Soto.

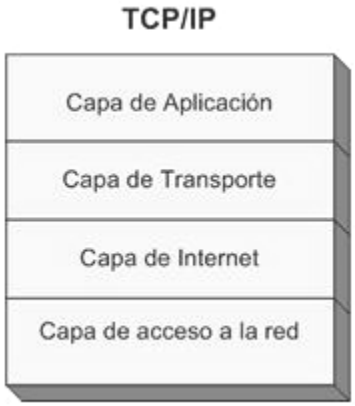


Figura 2.11 Capas del protocolo TCP/IP[27]

Capa de acceso a la red

Los protocolos de esta capa proporcionan al sistema los medios para enviar los datos a otros dispositivos conectados a la red. Es en esta capa donde se define como usar la red para enviar un datagrama. Es la única capa de la pila cuyos protocolos deben conocer los detalles de la red física.

Las principales funciones de los protocolos definidos en esta capa son:

- Encapsulación de los datagramas dentro de los marcos a transmitir por la red.
- Traducción de las direcciones IP a las direcciones físicas de la red.

Capa de Internet

Determina la ruta que tendrán los paquetes desde su origen hasta su destino sin importar que éstos se encuentren en redes geográficamente distintas, el protocolo específico de esta capa es mediante la IP.

Capa de transporte

Esta capa está definida por dos protocolos principales **TCP** (Protocolo de Control de Transmisión) el cual es un protocolo orientado a conexión que asegura la fiabilidad de un transferencia ya que verifica que los paquetes sean re ensamblados en el orden en el que éstos fueron enviados, y **UDP** (Protocolo de Datagrama de Usuario) el cual es un protocolo no orientado a conexión utilizado en consultas únicas de solicitud/respuesta arquitectura cliente/servidor en las cuales la velocidad de envío de información es más importante que la precisión de la información enviada.

[27] http://www.textoscientificos.com/redes/tcp-ip/comparacion-modelo-osi
TCP/IP y el modelo OSI

Capa de aplicación

Permite la interacción entre el usuario y la información en la red gracias a la utilización de aplicaciones que trabajan con los protocolos de nivel más alto (**HTTP, SSH, FTP, SMTP, POP3**, etc.). Además la capa de aplicación sincroniza la transmisión de datos entre aplicaciones y protocolos y verifica la disponibilidad de los recursos necesarios para el inicio de sesiones.

Esquema de comunicación del protocolo TCP/IP

A continuación se explica el proceso que se da para establecer comunicación entre dos equipos mediante el protocolo TCP/IP y se muestra en la figura 2.12.

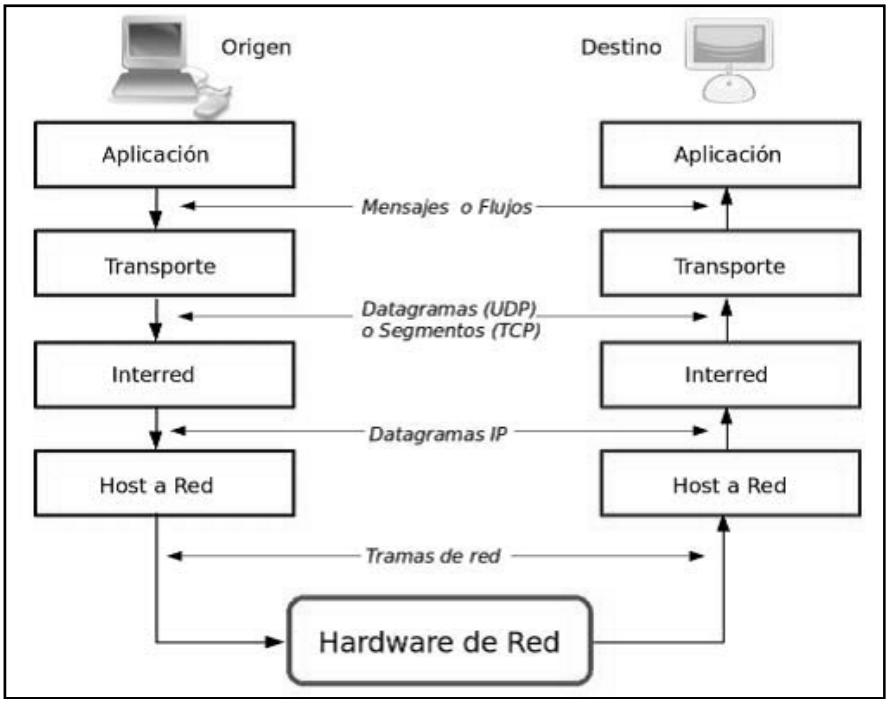


Figura 2.12 Ejemplo de comunicación utilizando el modelo de referencia TCP/IP.[28]

Proceso de comunicación

1. En la computadora origen, la capa de aplicación envía un flujo de bytes a la capa de transporte.[29]

2. La capa de transporte divide el flujo de bytes en segmentos TCP, asigna un encabezado con su número de secuencia a cada uno de los segmentos generados y envía los segmentos a la capa de internet.

3. La capa de internet crea un paquete con los datos que contienen los segmentos TCP a la cual le añade un encabezado que indica las direcciones IP de origen y de destino. También se determina

[28] Gómez Macías, 2008, Diseño de un cluster Beowulf para realizar cálculos Monte Carlo.
[29] Idem.

la dirección física de la computadora destino. Se envía el paquete y la dirección física a la capa de *host* a red.

4. La capa de host a red transmite el paquete.

5. Cuando se alcanza la computadora destino, la capa de *host* a red descarta el encabezado de enlace y envía el paquete IP a la capa de internet.

6. La capa de internet verifica el encabezado del paquete. Si la suma de comprobación del encabezado no coincide con la calculada, el paquete se desecha.

7. Si las sumas coinciden, la capa de internet descarta el encabezado y envía los segmentos a la capa de transporte.

8. La capa de transporte calcula una suma de comprobación, si la suma no coincide con la suma transmitida en el encabezado entonces el segmento es descartado. Si la suma coincide y el segmento está en la secuencia correcta la capa de transporte envía un reconocimiento a la computadora destino.

9. La capa de transporte descarta el encabezado TCP y transfiere los bytes del segmento que acaba de recibir a la capa de aplicación.

10. La capa de aplicación de la computadora destino recibe el flujo de bytes como si estuviera conectado directamente a la aplicación de la computadora origen.

Capa del sistema operativo

Es el programa que oculta al usuario los complejos procesos que requiere el control del *hardware* subyacente de una computadora y le presenta solamente una interfaz sencilla orientada hacia el uso de archivos. Además el sistema operativo realiza un reparto ordenado y controlado de los recursos del sistema entre el *hardware* y *software* que compiten por obtenerlos.

GNU Linux

GNU/Linux es, a simple vista, un sistema operativo. Es una implementación de libre distribución UNIX para computadoras personales (pc), servidores, y estaciones de trabajo. [30]

Como sistema operativo, GNU/Linux es muy eficiente y tiene un excelente diseño. Es multitarea, multiusuario, multiplataforma y multiprocesador; en las plataformas Intel corre en modo protegido; protege la memoria para que un programa no pueda hacer caer al resto del sistema; carga sólo las partes de un programa que se usan, comparte la memoria entre programas aumentando la velocidad y disminuyendo el uso de memoria, usa un sistema de memoria virtual por páginas; utiliza toda la memoria libre para cache, se distribuye con código fuente y soporta redes tanto en TCP/IP como en otros protocolos.

[30] http://www.grulic.org.ar/linux.html
Linux Information Sheet 1997 por Michael K. Johnson

Los componentes de un sistema GNU/Linux no están en el dominio público, ni son ***shareware*** (modalidad de distribución de *software* de forma gratuita para que el usuario pueda evaluarla), son lo que se llama "*software* libre". Esto significa que el código fuente está disponible a todo el que lo quiera y siempre lo estará. Todo esto está reglamentado por la *Licencia Pública General* ***GNU***. Esta licencia se encarga de que GNU/Linux permanezca siempre libre.

Todo esto resulta en un sistema de alta calidad tecnológica, con menos errores que los sistemas comerciales, a un costo cero o muy bajo y con la disponibilidad del código fuente que permite aprender, modificar o ayudar al desarrollo del sistema.

Historia de Linux, GNU y el *software* libre

En 1971, cuando Richard Stallman empezó su carrera en el **MIT** (Instituto Tecnológico de Massachusetts), el trabajaba en un grupo que usaba exclusivamente *software* libre. Hasta las grandes compañías distribuían *software* libre. Los programadores tenían la libertad de cooperar entre ellos y usualmente la ejercían.

Hacia la década del 80, la mayoría del *software* se había vuelto propietario, o sea, tenía dueños que prohibían y evitaban la cooperación entre los usuarios. Esto hizo que en 1983, Richard Stallman concibiera la *Free Software Foundation* (Fundación *software* libre, FSF) y en ésta *el proyecto GNU* como una forma de recuperar el espíritu cooperativo de los primeros días de la computación, y posibilitar nuevamente la cooperación sacando los obstáculos impuestos por los dueños del *software* propietario.

El proyecto *GNU* consiste en el desarrollo de un sistema operativo y juego de aplicaciones totalmente libre y compatible con ***UNIX***. El proyecto incluye desarrollar una versión libre de cualquier aplicación que no se disponga libre.

En 1990, se habían encontrado o escrito la mayoría de los componentes mayores del sistema operativo excepto uno: el ***kernel*** o núcleo. Para ese entonces, Linux comenzó como proyecto personal del entonces estudiante Linus Torvalds, que se basó en el ***Minix*** de Andy Tanenbaum (profesor que creó su propio clon de *UNIX* para usarlo en su docencia). Combinando Linux con el resto del sistema *GNU* se llegó a la meta inicial de un sistema operativo libre: El sistema GNU basado en Linux.

Requerimientos de *hardware* para correr GNU/Linux

Debido a su eficiente aprovechamiento de recursos, GNU/Linux tiene requisitos de *hardware* mínimos muy bajos: Una configuración mínima puede ser con 1MB de RAM, y una unidad de cd, más teclado, tarjeta de vídeo, monitor, etc., esto es suficiente para arrancar y entrar al sistema.

Para tener un sistema con todos los comandos importantes y una o dos aplicaciones pequeñas se requieren alrededor de 10 MB de disco duro.

GNU/Linux frente a los otros sistemas operativos

GNU/Linux es una muy buena alternativa frente a los demás sistemas operativos. Más allá de las ventajas evidentes de costo, ofrece algunas características muy notables.

En comparación con las otras versiones de Unix, la velocidad y confiabilidad de GNU/Linux son muy superiores. También está en ventaja sobre la disponibilidad de aplicaciones.

Comparado con sistemas operativos como los de Microsoft Windows, GNU/Linux también sale ganando. Los bajos requisitos de *hardware* permiten hacer un sistema potente y útil. Esta misma característica permite aprovechar al máximo las capacidades de las computadoras más modernas.

Ventajas de utilizar GNU Linux en *clusters*

En sus orígenes fue utilizado principalmente por ser un sistema operativo que permitía realizar toda clase de modificaciones sin el riesgo de ser demandado por una compañía.

En la actualidad GNU Linux presenta varias características extra que son deseables en el funcionamiento de un *cluster* de alto rendimiento:

- Confiabilidad: GNU Linux tiene tras de sí más de 30 años de desarrollo en UNIX, el sistema operativo con la reputación de ser el más confiable de todos. Esta más que comprobado que cuando un servidor está bien configurado pueden pasar varios años sin tener que reiniciarlo.

- Seguridad: Este tema fue considerado desde el principio en GNU Linux y se comprueba al revisar el ***kernel***. Este diseño seguro desde adentro (y no implementado como adicional al sistema operativo) hace prácticamente inmune a GNU Linux de virus y otros tipos de amenazas.

- Funcionalidad: GNU Linux tiene gran capacidad para trabajar en ambientes de red y de permitir realizar configuraciones que no necesitan del reinicio total del sistema. Además la administración de usuarios es sencilla y rápida.

- Desempeño: Solo basta con revisar la cantidad de supercomputadoras que trabajan con GNU Linux en la actualidad y se comprobará el grado de desempeño superior que puede llegar a brindar este sistema operativo cuando está bien configurado.

- Costo: La gran mayoría de las distribuciones de GNU Linux son gratuitas, normalmente el único costo está en el medio en el que se distribuye (CD, DVD) o del servicio de Internet que se tenga contratado para poder descargarlo.

Capa de *middleware*

Es el ***software*** intermediario que se sitúa entre una aplicación y un sistema operativo y permite a los procesos que se ejecutan en una o más computadoras interactuar entre sí a través de una red de datos.

Capa de aplicación

Se refiere al conjunto de programas que realizan sus funciones directamente para el usuario, en el caso de los *clusters* de alto rendimiento es el programa que se paraleliza para así obtener resultados en un menor tiempo.

2.4 Replicación con PostgreSQL

La replicación es uno de los métodos clave para asegurar la disponibilidad de los datos dentro de las bases de datos, la replicación es segura y útil.

Para entender la replicación y balanceo de carga, es necesario saber sobre la funcionalidad de un sistema gestor de bases de datos (DBMS). A continuación se presentan el modelo relacional de bases de datos y sus características.

2.4.1 Bases de datos relacionales

Las primeras bases de datos electrónicas fueron construidas en los años 60, el sistema más común en aquel momento fue llamado **Codasyl**, un proyecto sostenido por el departamento de defensa de los E.E.U.U. También IBM tenía su propia puesta en práctica llamada **IMS**. [31]

En 1970, un investigador de IBM llamado Edgar Frank Codd desarrolló un modelo matemático para almacenar datos.

Frank Codd se percató de que existían bases de datos en el mercado las cuales decían ser relacionales, pero lo único que hacían era guardar la información en las tablas, sin estar literalmente normalizadas; entonces éste publicó 12 reglas que un verdadero sistema relacional debería tener:

Regla 0

El sistema debe ser relacional, base de datos y administrador de sistema. Ese sistema debe utilizar sus facilidades relacionales (exclusivamente) para manejar la base de datos.

Regla 1

La regla de la información, toda la información en la base de datos es representada unidireccionalmente, por valores en posiciones de las columnas dentro de filas de tablas.

Regla 2

La regla del acceso garantizado, todos los datos deben ser accesibles sin ambigüedad. Esta regla es esencialmente una nueva exposición del requisito fundamental para las llaves primarias. Dice que cada valor escalar individual en la base de datos debe ser lógicamente direccionable especificando el nombre de la tabla, la columna que lo contiene y la llave primaria.

Regla 3

El tratamiento sistemático de valores nulos, el sistema de gestión de base de datos debe permitir que haya campos nulos. Debe tener una representación de la "información que falta y de la información inaplicable" que es sistemática, distinto de todos los valores regulares.

[31] Mikko Partio, 2007, Evaluation of PostgreSQL replication and load balancing implementations.

Regla 4

Catálogo dinámico en línea basado en el modelo relacional, el sistema debe soportar un catálogo en línea, el catálogo relacional deber ser accesible a los usuarios autorizados. Es decir, los usuarios deben poder tener acceso a la estructura de la base de datos (catálogo).

Regla 5

La regla comprensiva del sublenguaje de los datos, el sistema debe soportar por lo menos un lenguaje relacional que:

- Tenga una sintaxis lineal.
- Puede ser utilizado recíprocamente y dentro de programas de uso.
- Soporte operaciones de definición de datos, operaciones de manipulación de datos (actualización así como la recuperación), seguridad e integridad y operaciones de administración de transacciones.

Regla 6

Regla de actualización, todas las vistas que son teóricamente actualizables deben ser actualizables por el sistema.

Regla 7

Alto nivel de inserción, actualización, y cancelación, el sistema debe soportar suministrar datos en el mismo tiempo que se inserte, actualiza o este borrando. Esto significa que los datos se pueden recuperar de una base de datos relacional en los sistemas construidos de datos de filas múltiples y/o de tablas múltiples.

Regla 8

Independencia de datos físico, los cambios en el nivel físico (cómo se almacenan los datos, si en arreglos o en las listas encadenadas los etcétera.) no debe requerir un cambio a una solicitud basada en la estructura.

Regla 9

Independencia de datos lógica, los cambios al nivel lógico (tablas, columnas, filas, etcétera) no deben requerir un cambio a una solicitud basada en la estructura. La independencia de datos lógica es más difícil de lograr que la independencia física de datos.

Regla 10

Independencia de la integridad, las limitaciones de la integridad se deben especificar por separado de los programas de la aplicación y se almacenan en la base de datos. Debe ser posible cambiar esas limitaciones sin afectar innecesariamente las aplicaciones existentes.

Regla 11

Independencia de la distribución, la distribución de las porciones de la base de datos a las varias localizaciones debe ser invisible a los usuarios de la base de datos. Los usos existentes deben continuar funcionando con éxito:

1. Cuando una versión distribuida del **SGBD** se introdujo por primera vez
2. Cuando se distribuyen los datos existentes se redistribuyen en todo el sistema.

Regla 12

La regla de la no subversión, si un sistema relacional tiene un lenguaje de bajo nivel, ese bajo nivel no puede ser usado para saltarse (subvertir) las reglas de integridad y los limitantes expresados en los lenguajes relacionales de más alto nivel.

En aquel momento los trabajos de investigación de F. Codd fueron ignorados principalmente por la comunidad técnica porque el *hardware* con el que se contaba no resultaba lo suficientemente eficiente para soportar la aplicación basada en el complejo modelo.

Algunos años más adelante dos proyectos comenzaron a implementar sistemas de bases de datos relacionales basados en el modelo de Codd, el sistema de IBM llamado R y el sistema llamado Ingres de la Universidad de Berkeley. El sistema R de de IBM no fue exitoso, pero se logró desarrollar el lenguaje de alto nivel que necesitaba el modelo de Codd. El lenguaje desarrollado fue llamado SEQUEL (*Structured English Query Language*), que posteriormente fue abreviado como SQL.

El proyecto de Ingres tenía mayor éxito, y sostenido por los militares que desarrollaron una puesta en práctica bastante exacta del modelo original de Codd. La mayor parte de los sistemas de gestión de bases de datos modernos están conectados de alguna manera con el proyecto original de Ingres.

Una base de datos relacional consiste en relaciones, y una relación se puede considerar de forma lógica como conjuntos de datos llamados ***"tuplas"*** o también puede entenderse como una tabla con columnas y filas.

Un modelo de datos es llamado esquema, y en él se definen, el nombre de las relaciones, el nombre de la tabla y el tipo de atributo.

En consecuencia las *tuplas* en una relación deben satisfacer los requerimientos del esquema que se ha definido, además de cumplir la condición de unicidad (que las *tuplas* sean únicos e irrepetibles) definida por la constante de integridad.

Las ventajas del modelo relacional sobre el modelo jerárquico son incomparables y radica en un mejor funcionamiento, se puede hacer fácilmente escalable, fácil de implementar nueva tecnología de *hardware* y el hecho de que sea muy flexible ofrece una solución para cada tipo de dato existente.

2.4.2 SQL

El lenguaje de consulta estructurado (**SQL**) es un lenguaje de base de datos normalizado, utilizado por los diferentes motores de bases de datos para realizar determinadas operaciones sobre los datos o sobre la estructura de los mismos. Fue originalmente desarrollado por el equipo de IBM mediante el proyecto R, más tarde fue estandarizado por la organización **ISO/ANSI**.[32]

En 1986, el ANSI adoptó SQL como estándar para los lenguajes relacionales y en 1987 se transformó en estándar ISO. Esta versión del estándar va con el nombre de SQL/86. En los años siguientes, éste ha sufrido diversas revisiones que han conducido a la versión actual.

El hecho de tener un estándar definido por un lenguaje para bases de datos relacionales abre potencialmente el camino a la intercomunicabilidad entre todos los productos que se basan en él.

Componentes de SQL

El lenguaje SQL está compuesto por comandos, cláusulas, operadores y funciones de agregado. Estos elementos se combinan en las instrucciones para crear, actualizar y manipular las bases de datos.

Comandos

Existen dos tipos principales de lenguajes de SQL:

- Los **DDL** (*Data Definition Language*) que permiten crear y definir nuevas bases de datos, campos e índices, tales como los que se muestran en la tabla 2.4:

Tabla 2.4 Comandos DDL

Comando	Descripción
CREATE	Utilizado para crear nuevas tablas, campos e índices.
DROP	Empleado para eliminar tablas e índices.
ALTER	Utilizado para modificar las tablas agregando campos o cambiando la definición de los campos.

- Los **DML** (*Data Manipulation Language*) que permiten generar consultas para ordenar, filtrar y extraer datos de la base de datos, como por ejemplo los que se muestran en la tabla 2.5:

Tabla 2.5 Comandos DML

Comando	Descripción
SELECT	Utilizado para consultar registros de la base de datos que satisfagan un criterio determinado.

[32] http://www.htmlpoint.com/sql/sql_04.htm
Breve historia de SQL.

INSERT	Utilizado para cargar lotes de datos en la base de datos en una única operación.
UPDATE	Utilizado para modificar los valores de los campos y registros especificados.
DELETE	Utilizado para eliminar registros de una tabla de una base de datos.

SQL también define el proceso de unión de dos o más tablas, permite *triggers*, manejo de transacciones, lenguaje orientado a objetos, manejo de información espacial, entre otras características. El lenguaje de SQL ofrece muchas opciones para modificar y trabajar con los datos de las bases de datos.

2.4.3 PostgreSQL

PostgreSQL es un gestor de bases de datos basado en el modelo relacional, aunque incorpora algunos conceptos del modelo Orientado a Objetos, tales como la herencia. Usa un subconjunto ampliado de SQL como lenguaje de consulta y está implementado siguiendo la arquitectura cliente/servidor. PostgreSQL ofrece gran variedad de herramientas y librerías para acceder a las bases de datos.

Historia de PostgreSQL

PostgreSQL comienza como un proyecto de investigación del grupo del Profesor Michael Stonebraker en Berkeley. Fue derivado del proyecto Postgres (*post-ingres*), que usaba un lenguaje de consulta más avanzado: POSTQUEL.

La implementación de Postgres DBMS comenzó en 1986, y no hubo una versión operativa hasta 1987. La versión 1.0 fue liberada en Junio de 1989 a unos pocos usuarios, tras la cual se liberó la versión 2.0 en Junio de 1990 debido a unas críticas sobre el sistema de reglas, que obligó a su reimplementación. La versión 3.0 apareció en el año 1991, e incluyó una serie de mejoras como una mayor eficiencia en el ejecutor de peticiones. El resto de versiones liberadas a partir de entonces, se centraron en la portabilidad del sistema. El proyecto se dio por finalizado con la versión 4.2.[33]

En 1994, Andrew Yu y Jolly Chen añadieron un intérprete de SQL a este gestor. Postgres95, como así se llamó fue liberado a Internet como un proyecto libre (*OpenSource*). Estaba escrito totalmente en C, y la primera versión fue un 25% más pequeña que Postgres, y entre un 30 y un 50% más rápida.

En 1996, los desarrolladores decidieron cambiar el nombre al DBMS, y lo llamaron PostgreSQL (versión 6.0) para reflejar la relación entre Postgres y las versiones recientes de SQL. Se crearon nuevas mejoras y modificaciones, que repercutieron en un 20-40% más de eficiencia, así como la incorporación del estándar SQL92.

La versión que se ofrece en la página oficial de PostgreSQL a fechas de éste escrito, es la versión 8.4.1.

[33] http://www.netpecos.org/docs/mysql_postgres/x15.html
PostGreSQL vs. MySQL.

Evolución cronológica de PostgreSQL

- **Ingres** (1977-1985) Universidad de Berkeley
- **Postgres** (1986–1994) ***Post***erior a In***gres***, Michael Stonebraker
- **Postgres95** (1995) Andrew Yu & Jolly Chen publican el código en Internet.
- **PostgreSQL 6** (1996–1999) PostgreSQL Global Development Group
- **PostgreSQL 7** (1999–2004) Foreing Key, mejoras en el rendimiento, etc.
- **PostgreSQL 8** (2005)[34]

En la figura 2.13 se muestra la línea cronológica de PostgreSQL:

Figura 2.13 Evolución cronológica de PostgreSQL.

Características de PostgreSQL

● Corre en casi todos los principales sistemas operativos: Linux, Unix, Solaris, Windows, etcétera.

● Soporte de todas las características profesionales de una base de datos seria (objetos de uso común, orientación a objetos, eventos en los objetos de la base de datos, etc.).

● Soporte para los lenguajes más populares del medio: PHP, C, C++, Java, Perl, Python, etc.

● Drivers: Odbc, Jdbc, .Net, etc.

● Soporte de protocolo de comunicación encriptado por SSL

● Proyectos de interfaces de administración WEB.

● Proyectos para datos geográficos/geométricos (PostGIS)

● Proyectos para soportar diversos lenguajes de programación como lenguajes de funciones internas del motor (pl/Php, pl/Java, pl/Python, etc.)

● Licencia BSD, la más permisiva de todas para los negocios

● Versiones comerciales derivadas del proyecto libre.

[34] http://www.apesol.org.pe
Asociación Peruana de Software Libre.

Limitantes de PostgreSQL 8.2.4

En la tabla 2.6 se muestran las limitantes que tiene la versión 8.2.4 de PostgreSQL.

Tabla 2.6 Características de PostgreSQL 8.2.4

Característica	**Limitante**
Máximo de bases de datos	Ilimitado
Máximo de tamaño de tabla	32TB
Máximo de tamaño de registro	1.6TB
Máximo de tamaño de campo	1GB
Máximo de registros por Tabla	Ilimitado
Máximo de campos por tabla	250 a 1600 (depende de los tipos usados)
Máximo de índices por tabla	Ilimitado
Número de lenguajes en los que se puede programar funciones	pl/pgsql, java, perl, python, php, C, C++

2.4.4 Replicación

La replicación es un método para asegurar que la información contenida dentro de las bases de datos se encuentre segura y disponible, si ocurriera una falla en un nodo. [35]

La replicación en el contexto de bases de datos, significa la distribución de la información en dos o más instancias de bases de datos, que idealmente tienen el mismo contenido todo el tiempo. Se utiliza para asegurar la consistencia de los datos entre recursos redundantes, en tanto en los componentes de *software* como en los de *hardware*, con el objetivo de mejorar la confiabilidad, la tolerancia a fallos y la disponibilidad.

[35] Michal Valenta,2008, Mechanism of replication whit Slony-I

Principales mecanismos de replicación con PostgreSQL

Sistemas de respaldo

La manera más sencilla de replicar, es simplemente haciendo un respaldo (*backup*) de vez en cuando de el nodo maestro, usando herramientas directamente de PostgreSQL, como ***pg_dump*** **y** ***pg_restore***. El *dump* del nodo maestro se considera terminado cuando todos los cambios en los datos hasta el más mínimo han sido respaldados.

PGCluster

PGCluster es un sistema de replicación que es síncrono y multi-maestro, garantiza la consistencia de los datos (cada transacción se termina completamente o ni siquiera se inicia) y permite total acceso a los múltiples nodos maestros.

Con una infraestructura de servidores un poco más complicada (balanceador de cargas, *cluster* de bases de datos y un servidor de replicación) hace posible tener escenarios con balanceo de cargas y alta disponibilidad. Además PGCluster soporta la última versión estable de PostgreSQL.

Slony-I

Slony es un sistema de replicación asíncrona basado en ***triggers***, originalmente fue diseñado por Jan Wieck, ingeniero de *software* de la compañía *Afilias* que apoyó el desarrollo del sistema.

Al mismo tiempo que existían otros sistemas de replicación, Slony-I apareció como un nuevo concepto. Crea un mecanismo de replicación que soporta más de una versión de PostgreSQL, además que también brinda la oportunidad de empezar a funcionar sin necesidad de reiniciar o detener las bases de datos del sistema o sin la necesidad de utilizar los comandos ***dump*** y ***restore*** para las bases replicadas.

Slony-I soporta la replicación en cascada (maestro a múltiples esclavos) que es práctico para reducir la utilización del ancho de banda o reduce la carga de trabajo del nodo maestro del sistema. La replicación puede ser utilizada para crear alta disponibilidad en las bases de datos del *cluster* o para no tener tiempos fuera de servicio (*downtime)* en la migración a una nueva versión de PostgreSQL.

La configuración de un sistema de replicación con Slony-I se puede cambiar dinámicamente, pero Slony-I no posee ninguna funcionalidad que detecte la falla de un nodo del *cluster*, esto es necesario proveerlo con otro *software* que detecte y realice el cambio de rol de un nodo esclavo para funcionar como el nuevo nodo maestro.

Slony-II

Slony-II es más un concepto teórico que una implementación real. Pretende tener una replicación asíncrona que sobrepase a Slony-I y debe de soportar escenarios de replicación multi-maestro.

Técnicas de replicación

La replicación en el ámbito de las bases de datos, se entiende como la distribución de los datos en dos o más instancias de bases de datos y que idealmente contienen la misma información.

La replicación puede ser dividida en dos diferentes técnicas:

- Replicación síncrona
- Replicación asíncrona

Y a su vez, éstas dos técnicas pueden ser divididas en:

- Replicación maestro/esclavo
- Replicación multi-maestro

Replicación síncrona

La replicación síncrona garantiza la no pérdida posible de información debido a que las operaciones de escritura de datos deben estar completas en ambos nodos del *cluster* o en ninguno.

En la replicación síncrona el contenido de las bases de datos que se replican es siempre el mismo sin importar las circunstancias. La replicación síncrona significa que todos los nodos del *cluster* en el proceso de la replicación, deben de tener exactamente el mismo contenido de información todo el tiempo, todos los nodos actúan como si fueran “servidores maestros”.

Un servidor maestro (*master server*) es un término utilizado para decir que un nodo puede realizar operaciones de lectura y escritura a la base de datos.

Todas las consultas (***queries***) de lectura pueden ser realizadas por un solo nodo y las operaciones de escritura de datos a la base son distribuidas a todos los nodos participantes en la replicación, todas las operaciones de escritura son permanentemente confirmadas y puestas en línea, sólo cuando todos los nodos han confirmado que tienen la operación.

En la figura 2.14 se muestra un diagrama del funcionamiento de la replicación síncrona, multi-maestro.

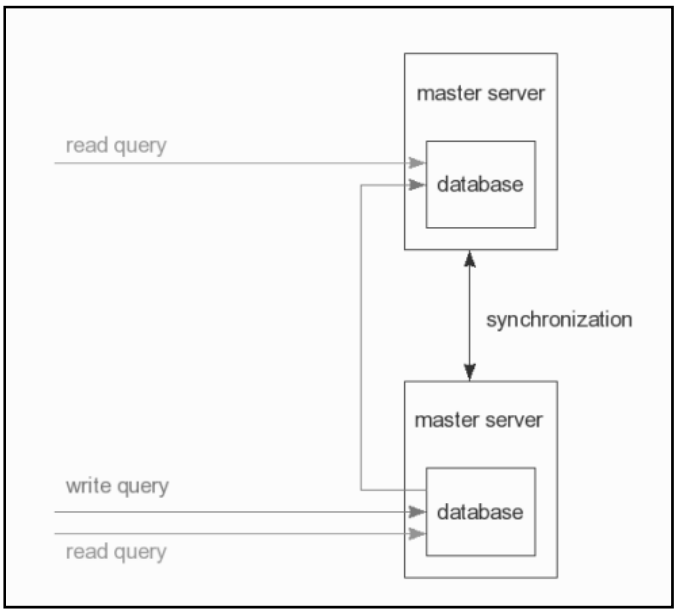


Figura 2.14 Replicación síncrona, multi-maestro. [36]

Replicación asíncrona

El segundo método de replicación es la llamada replicación asíncrona y consiste usualmente en un nodo maestro y uno o más nodos esclavos. Todas las operaciones de escritura son mandadas al nodo maestro el cual las procesa y envía los cambios a los nodos esclavos periódicamente.

Los nodos esclavos sólo pueden realizar operaciones de sólo lectura, durante el tiempo que toma al nodo maestro replicar los cambios a los nodos esclavos, las bases de datos no contienen la misma información.

La replicación asíncrona considera que las operaciones de escritura son finalizadas cuando la operación se termina en el nodo maestro principal, y el nodo remoto es actualizado también pero con cierto tiempo de retraso. Esta replicación puede tardar algunos segundos o varias horas, dependiendo la carga de la base de datos y el desempeño del *hardware* donde estén contenidas las bases de datos.[37]

La figura 2.15 muestra un ejemplo de sincronización asíncrona, maestro-esclavo.

[36] Mikko Partio, 2007, Evaluation of PostgreSQL replication and load balancing implementations.
[37] Idem.

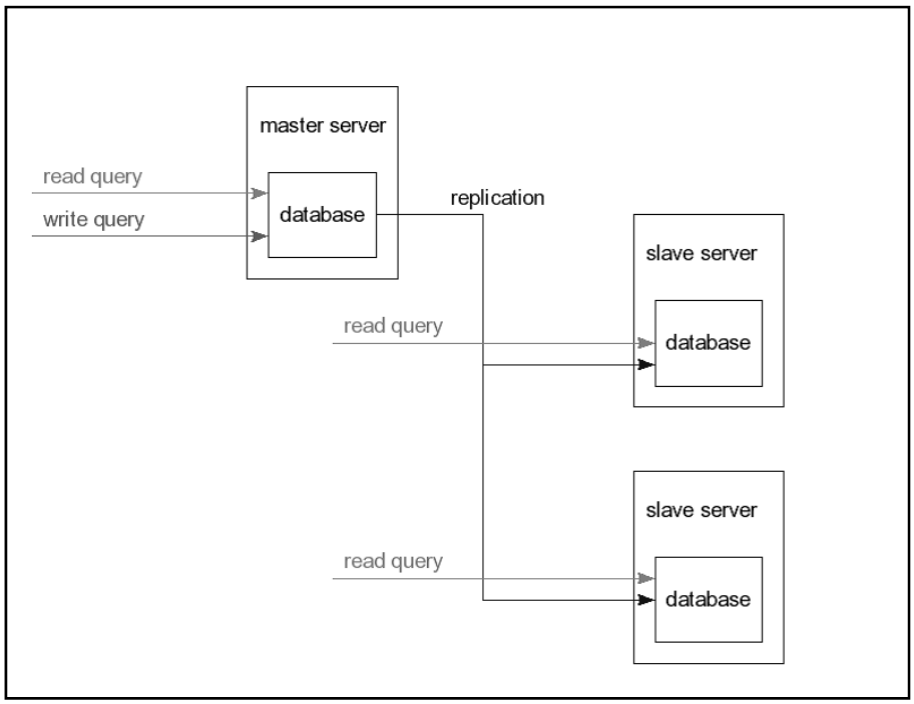


Figura 2.15 Replicación asíncrona, maestro-esclavo.[38]

Ventajas de la replicación

Replicación con alto desempeño

Las ventajas que representa la replicación se resumen en dos principales razones para utilizarla:

- Mejora el desempeño
- Incrementa la disponibilidad

La replicación de las bases de datos es usualmente implementada para incrementar el desempeño del sistema, tal esfuerzo se realiza típicamente cuando se incrementan las operaciones de lectura o escritura en las bases de datos.

La replicación de las bases de datos en un sistema, abarca como se había mencionado principalmente dos tipos de replicación, la replicación maestro-esclavo y la replicación multi-maestro que se abordarán a continuación.

Replicación maestro/esclavo

La replicación maestro/esclavo es probablemente la más común, en este escenario usualmente todas las operaciones son realizadas en el nodo maestro y las actualizaciones en los datos se transfieren transparentemente al nodo esclavo, que funciona como un respaldo, si algo sale mal en el nodo maestro, un nodo esclavo toma el rol del nodo maestro.

[38] Idem.

La replicación maestro/esclavo, como se aprecia en la figura 2.16, es bastante popular por que mejora el funcionamiento en las peticiones de lectura. En este tipo de escenario los nodos esclavos realizan las operaciones de sólo lectura y las actualizaciones son mandadas al nodo maestro. Mientras el nodo maestro pueda soportar el número de actualizaciones que se realizan, el sistema puede ser facilmente escalable de manera simple, sólo agregando nuevos nodos esclavos.[39]

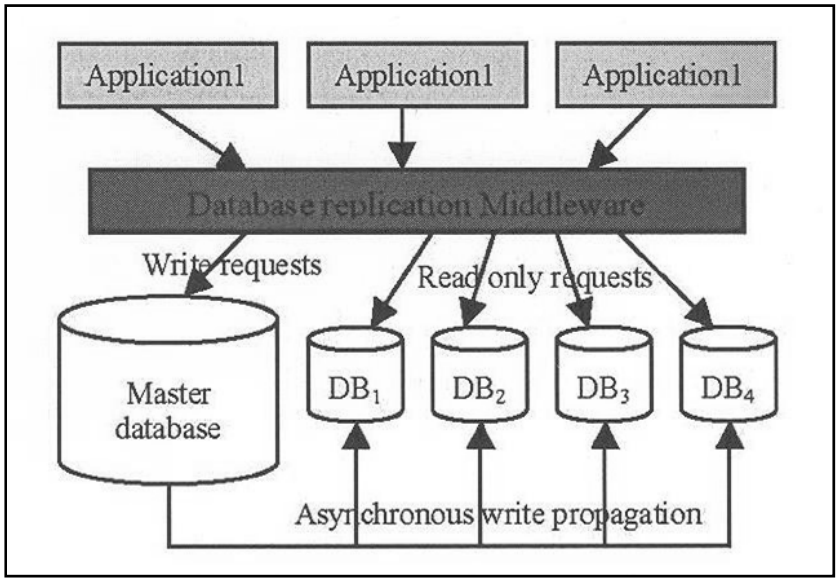


Figura 2.16 Replicación maestro/esclavo.[40]

Algunos ejemplos de *software* comercial para proveer replicación asíncrona, maestro-esclavo son:

- Microsoft SQL Server replication
- Oracle Streams
- Sybase Replication Server
- IBM DB2 Data Propagator
- Veritas Volume Replicator

Replicación multi-maestro

La replicación multi-maestro permite que cada réplica tenga una copia completa de la base de datos que realiza las peticiones de lecturas y escrituras. Sin embargo las réplicas necesitan sincronizarse en una serialización ordenada de las transacciones, por lo tanto, cada réplica ejecuta transacciones para mantenerse actualizada en el mismo orden. Esta replicación requiere un

[39] Cecchet, Ailamki, Candea, 2007, *Middleware*-based Database Replication: The Gaps between Theory and Practice.
[40] Idem.

sofisticado sistema transaccional el cual necesariamente debe evitar los problemas que se presenten en la actualización de los datos.[41]

También las transacciones concurrentes puedan presentar algún conflicto dejando algunas transacciones pendientes o abortándolas , lo cual limita la disponibilidad del sistema.

En la replicación multi-maestro se pueden realizar operaciones de actualización en cualquier nodo del *cluster*, porque todos se consideran nodos maestros, que pueden realizar operaciones de lectura y escritura, como se aprecia en la figura 2.17.

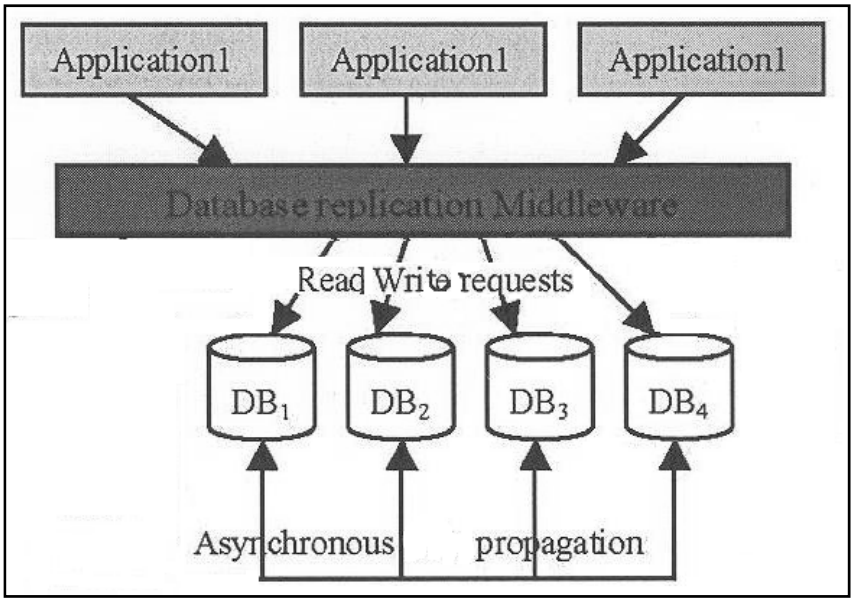


Figura 2.17 Replicación multi-maestro.[42]

Algunos ejemplos de *software* comercial de arquitecturas multi-maestro son:

- Continuent uni/cluster
- Avokia ap/live
- DB2 Integrated cluster

Replicación con alta disponibilidad

Si queremos hablar de alta disponibilidad, aplicamos este concepto cuando se tienen pequeños tiempos fuera de servicio (*downtime*). Cada tiempo de *downtime* puede ser planeado o no, dependiendo de lo que ocurra bajo el control del administrador o no. El *downtime* planeado generalmente sucede cuando se realizan operaciones de mantenimiento del *hardware* o del *software*, mientras que el *downtime* no planeado ocurre en cualquier momento y es causado por

[41] Cecchet, Ailamki, Candea, 2007, *Middleware*-based Database Replication: The Gaps between Theory and Practice.
[42] Idem.

fallas imprevistas de *hardware*, errores en el *software,* errores humanos entre otras posibles causas.[43]

Un sistema disponible se calcula aproximadamente con el promedio del tiempo total que se encuentra en línea o disponible.

La fórmula para calcular la disponibilidad es:

$$\text{Disponibilidad} = \frac{MTTF}{MTTF+MTTR} \longrightarrow \quad \text{No disponible} = \frac{MTTR}{MTTF+MTTR} \approx \frac{MTTR}{MTTF}$$

Donde:

MTTF(*Mean Time To Failure*) es el tiempo promedio de la falla

MTTR (*Mean Time To Repair*) es el tiempo promedio para reparar la falla

Como MTTF >>MTTR, podemos decir que la no disponibilidad es aproximadamente (promedio del tiempo total del *downtime*) MTTR/MTTF.

La meta de la replicación junto con la tolerancia a fallos (*failover/failback*) es reducir el MTTR el tiempo que tardará en arreglar el fallo del sistema y por consecuencia aumentar la dsiponibilidad y reducir el *downtime*.

Ahora se hablará del recurso más utilizado para la replicación de bases de datos, tanto en el área del *software* libre como del lado comercial, según la técnica a utilizar de un solo maestro o multi-maestro, la meta final sigue siendo la misma: proveer una recuperación rápida de la falla de un nodo. Un nodo responde todos las solicitudes de consultas (***queries***) y cuando ocurre una falla en dicho nodo se tiene que transferir su carga de trabajo de las solicitudes de consulta sobre el otro nodo.

En la figura 2.18 se muestra la puesta del *software* Slony–I con PostgreSQL, de manera que se obtenga un sistema con *failover*. Se entiende por ***failover*** la capacidad que posee un sistema de poder reubicar a los usuarios de una base de datos para ser cambiados a otro nodo de bases de datos que contenga una réplica de los datos sin importar en que nodo hubiera ocurrido la falla y todo esto de manera transparente para los usuarios de la base de datos. De manera diferente también existe el ***failback*** que ocurre cuando sucede una falla en la replica más reciente y entonces tiene que retroceder a la réplica anterior y por consecuencia los usuarios también son reubicados.

[43] Cecchet, Ailamki, Candea, 2007, *Middleware*-based Database Replication: The Gaps between Theory and Practice.

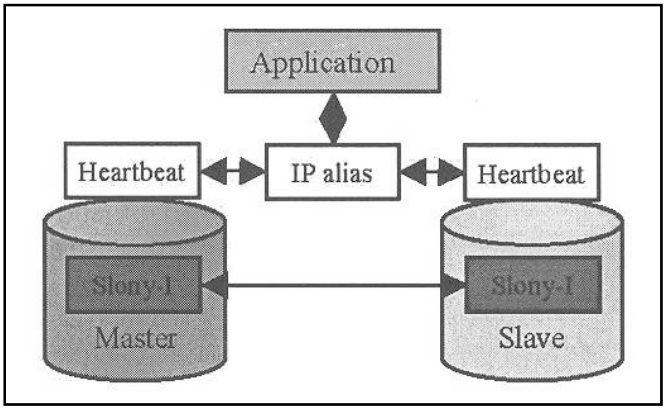


Figura 2.18 Failover con Slony-I.[44]

Aquí se tienen dos réplicas, una actuando como el maestro y la otra como esclavo , la aplicación se conecta con un balanceador de carga que redirecciona la petición al maestro y cuando una falla sea detectada en el nodo maestro via heartbeats, las peticiones son enrutadas al nodo esclavo. Ahora el nodo esclavo tiene las mismas capacidades que el nodo maestro, realiza operaciones de lectura y escritura como si fuera el maestro.

Tabla comparativa de *software* para implementar un cluster

En la tabla 2.7 se muestran las distintas opciones con las que se pueden implementar sistemas de alta diponibilidad mediante la replicación, donde se muestra tanto los *software* de licencia libre como los comerciales, así como sus principales características.

Tabla 2.7 Comparativa de *software*[45]

Programa	Licencia	Método de replicación	Tipo de sincronización	Pool de conexión	Balanceo de cargas
PGCluster	Libre	Multi-Maestro	Síncrona	No	Si
Slony-I	Libre	Maestro -Esclavo	Asíncrona	No	No
Bucardo	Libre	Multi-Maestro Maestro-Esclavo	Asíncrona	No	No
Londiste	Libre	Maestro -Esclavo	Asíncrona	No	No
Mammoth	Comercial	Maestro -Esclavo	Asíncrona	No	No
DB2 Cluster	Comercial	Multi-Maestro	Asíncrona	SI	SI

[44] Cecchet, Ailamki, Candea, 2007, *Middleware*-based Database Replication: The Gaps between Theory and Practice.

[45] http://www.postgresql.org.pe/articles/comparativa_dbms_libres.pdf
Comparativa de PostgreSQL vs otros DBMS Libres por Ernesto Quiñones.

Capítulo 3
Descripción del sistema

Capítulo 3 Descripción del sistema

3.1 Introducción

En el desarrollo del capítulo 3 se describirán los requerimientos necesarios en cuanto al diseño para realizar la implementación del *cluster* de alta disponibilidad de base de datos para la geDGAPA.

Cabe destacar la importancia que debe tener el diseño y la organización de un proyecto, ya que sin una adecuada organización de cada etapa de la implementación se pierde el orden de los tiempos de entrega derivándose en un aplazamiento indefinido del proyecto y una carga extra de trabajo innecesaria.

La organización correcta del proyecto permite la entrega en tiempo y forma adecuada a las necesidades planteadas originalmente.

3.2 Requerimientos funcionales del cluster

Los requisitos funcionales del *cluster* son el punto de inicio para una estrategia completa de desarrollo de soluciones. Para obtener los requisitos funcionales del sistema, se debe llevar a cabo un proceso de análisis del problema, el cual tiene como propósito fundamental obtener una descripción formal de la solución, que proporcione la información suficiente y necesaria para resolver el problema.

Es absolutamente necesario realizar un análisis del problema planteado, antes de hacer cualquier tarea de diseño del sistema. El éxito de un proyecto se basa principalmente en qué tan bien diseñado esté y para esto es fundamental analizar en forma detallada y crítica la serie de problemas que se presentan.

3.2.1 Planeación de un cluster

Un *cluster*, se considera como una solución, cuando es de vital importancia la información que se maneja y por la necesidad de tener aplicaciones que todo el tiempo estén en línea. Para poder realizar la planeación de un sistema de este tipo debemos tomar en cuenta los siguientes puntos[46]:

- **Conocer el ambiente del centro de datos.** Un *cluster* forma parte de un objetivo, el de mantener el sistema como de alta disponibilidad, los beneficios no se podrán alcanzar a menos de que éste haya sido hecho a la medida. Para que el ambiente se considere favorable debe incluir:

    - ✓ Un sistema de almacenamiento fuera del servidor.
    - ✓ Protección del sitio.

[46] Hernández Rodríguez, 2003, Implementación de un sistema de alta disponibilidad basado en tecnología de cluster.

- ✓ Fuente de poder ininterrumpible.
- ✓ Protección del servidor.
- ✓ Contar con refacciones.
- ✓ Tener un servicio de respuesta inmediata en caso de fallas.

También se debe considerar el tamaño y cuánto se piensa que crecerán los datos, asegurando que la energía eléctrica y el ancho de banda de la red sean lo suficientemente grande para satisfacer las necesidades futuras del servidor y de los sistemas de almacenamiento.

- **Funciones del negocio.** Para poder implementar una solución de alta disponibilidad satisfactoriamente, es útil realizar un análisis de los distintos niveles funcionales que comprenda el sistema y los componentes que integran cada nivel.

  - ✓ Categorizando esencialmente cuáles son las funciones de los negocios y estudiando cada uno de ellos para observar en qué ambiente se desarrolla de mejor manera.
  - ✓ Identificando los componentes de *hardware* que contribuyen para poder completar cada función.
  - ✓ Determinando los componentes de *software* que contribuyen para poder completar cada función.
  - ✓ Examinar varias opciones para poder determinar el nivel apropiado de la solución de alta disponibilidad que se debe implementar, basado en la tecnología actual y tomando en cuenta la importancia de las funciones de los negocios.

- **Componentes del *hardware*.** Para seleccionar los componentes de *hardware* adecuados, se necesita adquirir la información necesaria de la tecnología que se desea tener para poder tomar la decisión correcta. La organización y protección de cada una de las piezas se debe considerar importante como si fuera uno de los elementos más críticos dentro del *cluster*.

También se debe examinar la naturaleza de la red y el ambiente de los negocios. Antes de tomar una decisión, se debe pensar en los siguientes cuestionamientos:

- ✓ ¿Cuánta productividad se ha perdido cada minuto que el sistema no está disponible?
- ✓ ¿Cuáles han sido los componentes que más han fallado en el pasado?
- ✓ ¿Qué tiempo se toma en promedio recuperarse de las fallas que se han presentado?
- ✓ ¿Cuál ha sido el esfuerzo de los administradores por tratar de resolver estos problemas?

- **Componentes de *software***. Además de las opciones que se tienen de *hardware* redundante, también se debe prestar atención a las aplicaciones de misión crítica que están relacionadas con el *software* como son:

  - ✓ Bases de datos
  - ✓ Correo electrónico
  - ✓ Servicios de impresión

Evaluando cada aplicación de *software* se determina su comportamiento y su campo de trabajo en un ambiente *cluster*, algunas de las aplicaciones de *cluster* que sean inteligentes se adaptarán de manera más rápida a este tipo de ambiente aprovechando todos los beneficios que nos brinda el mismo.

- Plan de implementación de un *cluster*. Para diseñar un ambiente de *cluster* y obtener una mayor eficiencia, es necesario crear un plan de implementación detallado. Durante el proceso de esta implementación, se necesita determinar:

  - ✓ Los requerimientos de los grupos de trabajo.
  - ✓ Los requerimientos de los servidores para sostener la carga de trabajo.
  - ✓ Contemplar los grupos de aplicaciones.
  - ✓ Modelar posibles fallas para prevenirlas.
  - ✓ Contar con todos los requerimientos para implementar la redundancia.

De manera general, podemos decir que los puntos tratados con anterioridad, pretenden crear un bosquejo de la planificación previa a la implementación de un *cluster*, así como de dar a conocer cuáles son los medios y los ambientes necesarios para obtener los beneficios que nos otorga una solución de este tipo.

3.3 Modelo del análisis

3.3.1 Slony-I

El desarrollo de este proyecto se enfoca en la construcción de un sistema maestro-esclavo mediante el *software* Slony-I, el cual da la posibilidad de replicar largas bases de datos en un número razonable de nodos esclavos.

El desarrollo de varios algoritmos que utiliza Slony-I para su funcionamiento se deben a Vadim Mikheev, quien utiliza los siguientes términos que se relacionan directamente con Slony-I y que además provienen del idioma ruso y significan lo siguiente:

- Slon significa elefante
- Slony es el plural de elefante
- Slonik significa pequeño elefante

Es indispensable conocer el funcionamiento del motor de Slony-I, con la finalidad de saber cómo se deben de realizar las configuraciones del *cluster*, de acuerdo a nuestras necesidades específicas.

Slony-I es un *software* diseñado para centros de información o ***sites*** de respaldo, en donde el modo normal de operación es que todos los nodos se encuentren disponibles todo el tiempo y protegidos de cualquier intromisión no permitida.[47]

Para entender de manera adecuada, este subcapítulo, se debe tener claros los siguientes conceptos, que se usarán con frecuencia:

[47] http://www.slony.info/documentation/slonyintro.html#INTRODUCTION
Slony-I 2.0.3_RC2 Documentation.

Cluster

En términos de Slony-I, un *cluster* es un set de instancias de bases de datos de PostgreSQL, donde la replicación ocurre entre esas bases de datos.

Nodo

Un nodo en Slony-I se considera como una base de datos de PostgreSQL que participa en la replicación. También se define como la combinación de una base de datos y un proceso slon.

En una simple configuración, todos los nodos tienen un ***path*** o ruta unos con otros. Cada *path* le dirá a cada proceso slon como se debe conectar con los otros nodos de las demás bases de datos.[48]

A continuación en la figura 3.1 se muestra un simple arreglo de nodos, se cuenta con cuatro nodos, cada uno con su base de datos y su proceso slon, así mismo, se observan las conexiones que existen entre los nodos.

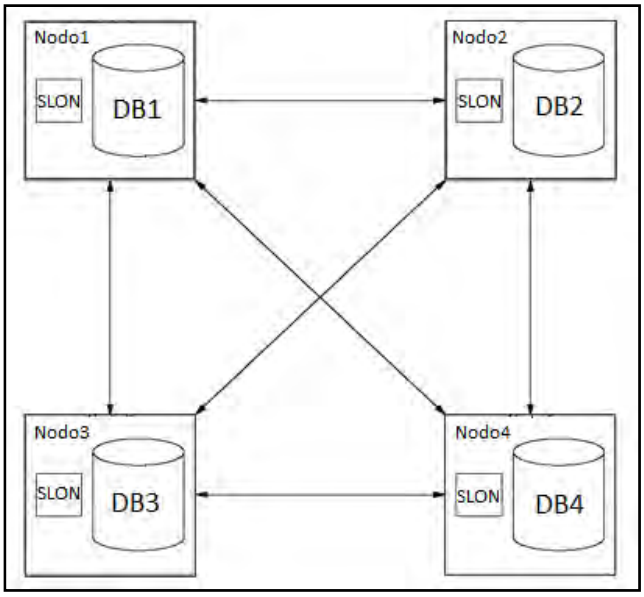


Figura 3.1 Red simple de nodos.

Set de replicación

Un *set* de replicación es definido como un conjunto de datos que son replicados entre los nodos del *cluster* de Slony-I.

Un *set* de replicación consiste en tener:

- Llaves en las tablas que serán las replicadas.
- Tablas que deben ser replicadas.
- Secuenciadores que deben ser aplicados.

[48] Jan Wieck, 2005, Slony-I System Overview.

Es probable que se tengan varios sets de replicación y es muy probable que el flujo de datos en el proceso de replicación no sea idéntico entre los sets de replicación, pero de manera que se obtengan los mismos resultados, es decir, todos los datos replicados en todos los nodos activos en el proceso de replicación.

Origen, proveedor y suscriptor

Cada set de replicación tiene un nodo origen, que es el único lugar donde el usuario tiene permitido modificar datos en las tablas que se están replicando, por ello, el nodo origen recibe también el nombre de nodo maestro y es el lugar principal de donde los datos provienen.

Los nodos subscriptores en el set de replicación, son los que quieren recibir los datos. El nodo origen o maestro nunca será considerado subscriptor. Pero Slony-I soporta el concepto de suscripciones conectadas en cascada, es decir, un nodo que se suscribe a un cierto set de replicación puede también comportarse como proveedor a otros nodos del *cluster* dentro del set de replicación.

Procesador de la información: **Slonik**

El procesador de información Slonik procesa los scripts que generan los eventos de replicación. Los scripts son para modificar la configuración del *cluster* de Slony-I, esto incluye, por ejemplo, agregar nodos, modificar nodos, eliminar nodos, modificar rutas de comunicación, etcétera.

Demonio: **Slon**

Para cada nodo en el *cluster*, habrá un proceso **slon** que manejará la actividad de la replicación del nodo. Slon es un programa desarrollado en lenguaje C que procesa eventos de replicación, estos eventos se clasifican en dos tipos:

- Eventos de Configuración. Estos normalmente ocurren cuando el script de slonik está corriendo y manda actualizaciones de la configuración del *cluster*.
- Eventos **SYNC**. Las actualizaciones a las tablas que son replicadas son agrupadas todas en eventos *SYNC*, que son grupos de transacciones que son aplicadas juntas a los nodos suscriptores o esclavos.

Teniendo los conceptos descritos previamente se puede proseguir a explicar como ocurre el funcionamiento interno de Slony-I, el cual implica un flujo de eventos entre los nodos.

La meta final de Slony-I es la replicación, pero para poner a trabajar la replicación es necesario entender otros factores del sistema y un factor importante, consiste en no relacionar un nodo cualquiera con un rol maestro o esclavo fijo por que estos roles pueden variar según la circunstancias que se presenten, es decir, si el nodo maestro falla entrará en su lugar un nodo esclavo que ahora será el maestro, por lo tanto, el rol que se tenía anterior a la falla claramente habrá cambiado.

En la tabla 3.1 podemos observar los diferentes cambios de rol que pueden ocurrir entre el nodo origen, proveedor y receptor, estos diferentes cambios de rol se les denominan eventos y se puede verificar su ocurrencia con el diagrama mostrado en la figura 3.2.

Tabla 3.1 Tabla de eventos

Nodo Origen	Nodo Receptor	Nodo Proveedor
1	2	1
1	3	2
1	4	3
2	1	2
2	3	2
2	4	3
3	1	2
3	2	3
3	4	3
4	1	2

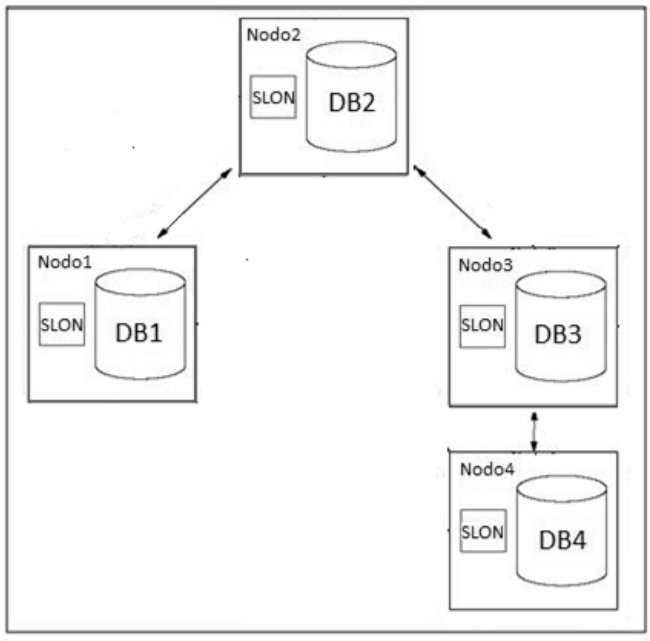


Figura 3.2 Diagrama de eventos.

Observado previamente los eventos que pueden ocurrir en los nodos del *cluster*, es necesario explicar cómo se realiza internamente estos cambios de roles, que deriva a su vez en un flujo de eventos entre los nodos.

El flujo de eventos ocurre de la siguiente manera:

1. Un evento ocurre en el nodo 3
2. El nodo 2 y el nodo 4 reciben la notificación, leen el evento, procesan y la confirman mediante una transacción en la base local.
3. El nodo 1 recibe la notificación, lee el evento en el nodo 2, procesa y confirma.
4. Cuando se está procesando la información mediante la transacción en el nodo 1, el nodo 2 y 4 realizan un *commit*, los ***threads*** remotos que escuchan los eventos, los notifican y propagan la confirmación del evento.
5. Periódicamente el *thread* de limpieza checa los eventos que han sido confirmados por todos los nodos y los remueve.

La figura 3.3 ejemplifica el flujo de eventos descrito previamente.

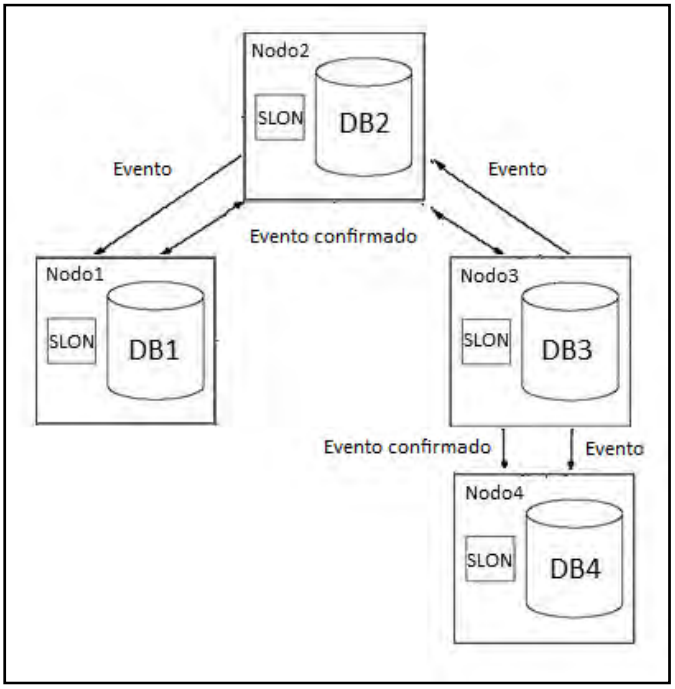


Figura 3.3 Flujo de un evento.

Slony-I maneja en su diseño ***sets*** y suscripciones, para organizar los objetos de una base de datos (tablas y secuencias), por otra parte para empezar a replicar datos, Slony-I necesita copiar una foto o vistazo inicial del set del nodo proveedor o maestro al nodo suscriptor o esclavo.

La figura 3.4 muestra como ocurre la suscripción de un *set* en el proceso de replicación.

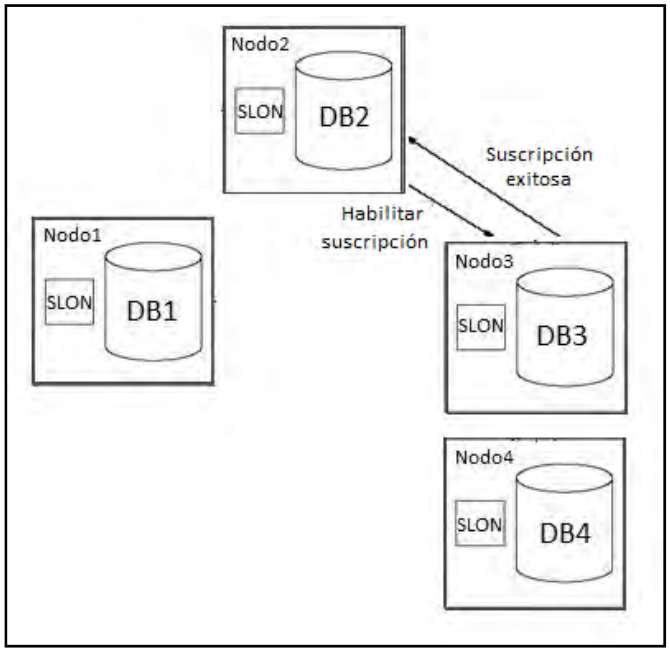


Figura 3.4 Suscripción de un set.

La siguiente secuencia asume que el nodo 2 es el origen del set, entonces:

1. El evento del set suscriptor es generado en el nodo 3 y como es normal el evento es propagado por todos los nodos.

2. Cuando el nodo 2 recibe el evento genera un evento de suscripción y lo envía de regreso.

3. Cuando el nodo 3 procesa el evento, lo permite y copia las tablas y secuencias para dicho set.

El resto de los nodos suscriptores también deben tener conocimiento del nuevo nodo suscriptor.

Maestro – esclavo en cascada

La estructura básica de un sistema utilizando Slony-I es la configuración de un maestro con uno o más esclavos. No todos los nodos esclavos recibirán la réplica directamente del nodo maestro, cada nodo recibe los datos de una fuente válida que es configurada para poder permitir el reenvió de los datos a otros nodos.

Hay tres puntos importantes detrás de esta capacidad para replicar, el primer punto a considerar es la escalabilidad, por ejemplo, la base de datos que reside en el nodo maestro recibe todas las operaciones de actualizaciones desde las aplicaciones del cliente, por lo tanto tiene una capacidad limitada para satisfacer las consultas que provienen de los esclavos, durante el proceso de replicación, por lo tanto es necesario considerar hasta que punto va a funcionar el arreglo de nodos que se posee y ver que tan escalable puede hacerse cuando sea necesario.

El segundo punto es limitar el ancho de banda utilizado para realizar el ***backup*** y mantener al mismo tiempo la disponibilidad de los múltiples nodos esclavos.

Y el último punto es construir escenarios ***failover***, en una replicación maestro con múltiples esclavos, es casi imposible que todos los nodos esclavos estén en el mismo avance de replicación cuando el nodo maestro falle, para asegurar que un nodo esclavo pueda ser promovido a nodo maestro es necesario, que todos los sistemas restantes estén de acuerdo sobre el *status* en el que se quedó la información. Como una transacción ***committed*** no puede ser cancelada o regresada al estado en el que se encontraba antes de empezar la transacción, el ***rolled back***, es indudablemente el estado más reciente de la sincronización de todos los nodos esclavos restantes.

Suscripción en cascada

Para mantener la copia inicial de un *set* desde el nodo origen, cada suscriptor tiene las instrucciones de copiar los datos desde un suscriptor existente, actuando posteriormente a la copia como proveedor de datos para un nuevo nodo suscriptor a él, como se aprecia en la figura 3.5.

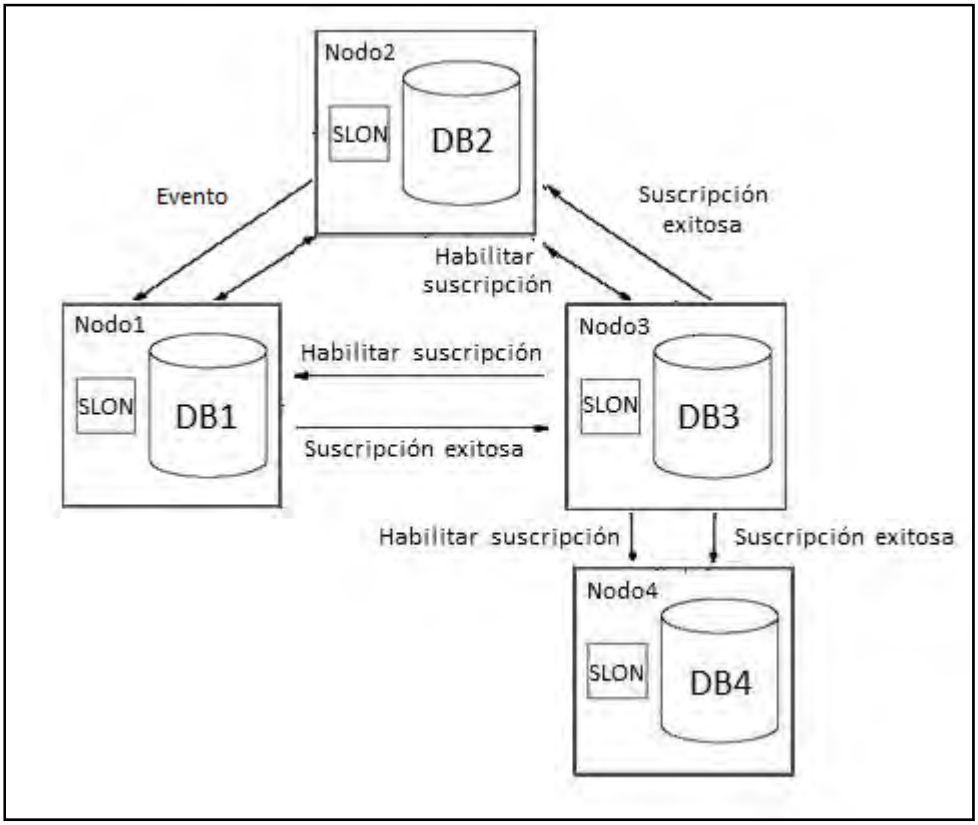


Figura 3.5 Suscripción en cascada.

Replicación de los datos

Una vez que se recibe un evento *SYNC* el nodo checa si está actualmente suscrito al set de replicación origen que manda el evento, si no se encuentra suscrito solo verifica el evento como cualquier otro sin realizar nada y termina con él. En cambio si se encuentra suscrito al *set* o más *sets* que provengan del *set* de replicación origen, procede a trabajar en la replicación, en base a los siguientes pasos.

1. El nodo checa que tenga conexiones con todos los nodos proveedores o esclavos que proporcionen replicación de la información a cualquier set de replicación que esté suscrito al evento *SYNC* de origen.[49]

[49] Jan Wieck, Slony-I a replication system for PostgreSQL.

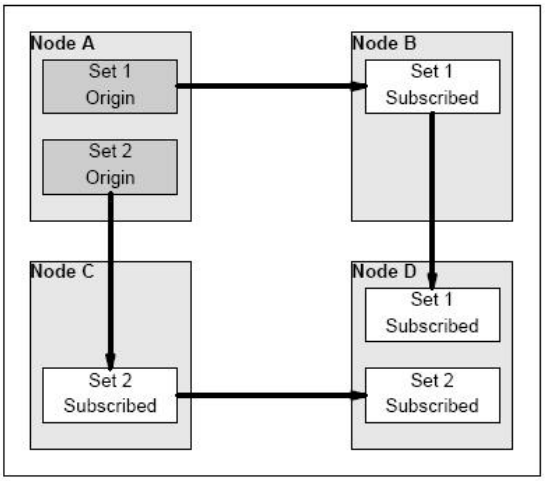


Figura 3.6 Replicación de sets.[50]

La figura 3.6 muestra un escenario donde el nodo B es configurado para replicar solamente el *set* 1, asimismo el nodo C es configurado para replicar solamente el *set* 2. Para propósitos de balanceo de carga, el nodo D está suscrito a ambos *sets* 1 y 2, para mantener la carga de trabajo en el nodo maestro A lo más bajo posible, se replica el *set* 1 del nodo B y el *set* 2 del nodo C.

1. El evento *SYNC* generado en el nodo A es mandado a ambos sets de replicación y toda la información de ambos *sets* que se encontraba antes del evento *SYNC* es enviada al nodo D en una sola transacción. Así el nodo D sólo puede procesar y empezar la replicación hasta que se encuentre finalizado el evento *SYNC*.

2. Ahora lo que el nodo demonio Slon necesita es una selección lógica de ***logs*** activos de tablas de cada nodo cercano participante en el evento *SYNC* de origen y en orden secuencial. Los datos seleccionados son restringidos a las tablas contenidas en todos los *sets* proporcionados por el nodo específico y obligados a sobre escribir sobre el evento *SYNC* pasado y generar el actual.

3. Ahora todos los *sets* remotos de replicación son reflejados en el nodo destino de replicación y aplicados a la base de datos local. Los *triggers* definen que cualquier tabla que vaya hacer replicada sea deshabilitada durante todo el proceso entero del *SYNC*. Todas las acciones realizadas por *triggers* que tienen efecto en las tablas que participan en la replicación deben también replicar dichas acciones generadas por los *triggers*.

4. El evento *SYNC* que tenga un problema será reportado, realizará una acción de *committed* y se mandará una confirmación del error a los demás eventos para que tengan conocimiento de la falla.

Failover

Para guardar un *log* de datos sobre un nodo, es necesario configurar todos los nodos suscriptores de un *set*, hasta que todos hayan confirmado ser parte del evento *SYNC*, se cumple el

[50] Jan Wieck, Slony-I a replication system for PostgreSQL.

primer paso para poder cambiar el nodo maestro por un esclavo, lo que llamamos comúnmente como *failover*.[51]

Cambiar de nodo maestro o proveedor del *log* de datos significa simplemente volver a empezar desde un punto arbitrario en cualquier momento comunicándose mediante *triggers* y eventos.

Failover no es más que una secuencia lógica de sincronización con otros nodos, cambiando el origen del *set* de replicación y el nodo proveedor o maestro del *set*.

El proceso de *failover* ocurre como se muestra en la figura 3.7:

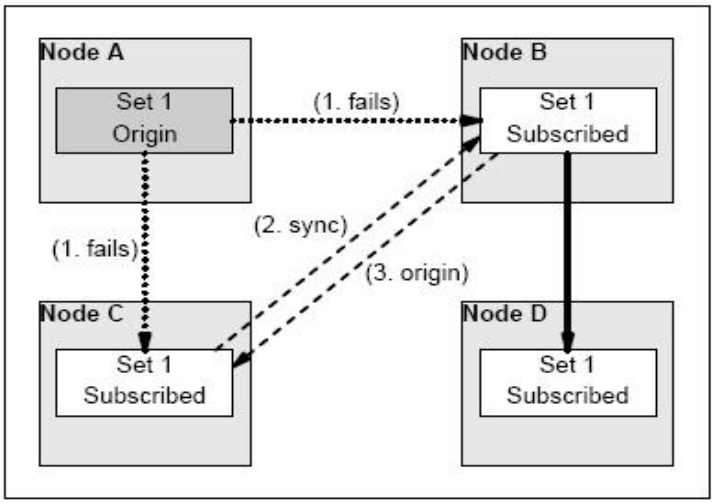


Figura 3.7 Failover.[52]

A continuación se explican los pasos del proceso de *failover*:

1. El nodo A falla, como se muestra en la figura 3.7, este es el origen de datos del *set* 1. El plan consiste en promover el nodo B como nodo maestro y dejar que el nodo C siga replicando con el nodo promovido como nodo maestro.

2. Como es probable que el nodo C tenga un avance mayor en su replicación que el nodo B, entonces, el nodo B primero preguntará por cada evento para ver si ya se encuentra replicado o no. Realmente no hay gran diferencia si se replicará directamente sobre nodo A.

3. Al mismo tiempo el nodo B seguramente tiene el mismo avance o un poco más que el nodo C, entonces el nodo B toma su lugar y se convierte en el nodo origen. El cambio del nodo proveedor hace que no se conozca si el nodo C haya replicado todos los eventos *SYNC* del nodo A, y de los que ya tenga replicados el nodo B. Entonces el evento origen proveniente del nodo B contendrá el último evento del nodo A conocido por el nodo B hasta ese momento, que debe de ser el último evento del nodo A conocido por todos los demás nodos del *cluster*. El proceso de cambio del evento origen sobre el nodo C es, que no puede ser confirmado hasta que el nodo C haya replicado todos los eventos del nodo A hasta el cambio del evento origen del nodo C. En ese

[51] Jan Wieck, Slony-I a replication system for PostgreSQL.
[52] Idem.

momento el nodo C es libre de continuar replicando usando el nodo B o D como su proveedor de datos.

Todo el proceso completo de *failover* se ve relativamente simple, sin embargo, la simplicidad tiene un precio, para este caso, ocurre cuando un nodo esclavo se vuelve no disponible, todos los demás nodos del *cluster* dejarán de limpiar y acumularán información de eventos y posibles *logs* de datos. Esto es importante porque si un nodo no está disponible por un tiempo largo, hay que cambiar la configuración y dejar que el sistema sepa que se utilizarán otras técnicas para reactivar el nodo, esto puede realizarse suspendiendo o desactivando el nodo de manera lógica o removiéndolo completamente de la configuración.

3.3.2 Seguridad

Firewall

Un *firewall* es un dispositivo que filtra el tráfico entre redes. El *firewall* puede ser un dispositivo físico o un *software* sobre un sistema operativo. En general debemos verlo como una caja con dos o más interfaces de red en la que se establecen una reglas de filtrado con las que se decide si una conexión determinada puede establecerse o no. [53]

Esa sería la definición genérica, hoy en día un *firewall* es un *hardware* específico con un sistema operativo o una **IOS** (*Internetwork Operating System*) que filtra el tráfico (TCP, UDP, ICMP, IP) y decide si un paquete pasa, se modifica, se convierte o se descarta. Para que un *firewall* entre redes funcione como tal debe tener al menos dos tarjetas de red. En la figura 3.8 se observa la topología clásica de un *firewall*:

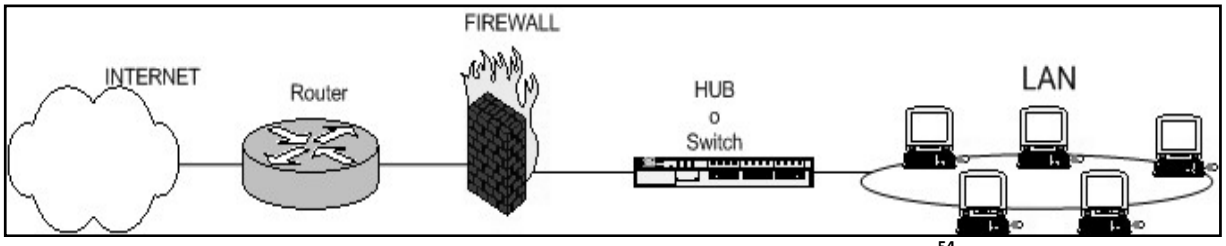


Figura 3.8 Esquema de un firewall típico entre red local e internet.[54]

Se observa un típico *firewall* para proteger una red local conectada a internet a través de un ***router***. El *firewall* debe colocarse entre el *router* (con un único cable) y la red local (conectado al ***switch*** o al ***hub*** de la LAN).

Hay dos maneras de implementar un *firewall*:

1) Política por defecto ACEPTAR: en principio todo lo que entra y sale por el *firewall* se acepta y solo se denegará lo que se diga explícitamente.

[53] Altardill Izura, IPTABLES Manual práctico.
[54] Idem.

2) Política por defecto DENEGAR: todo está denegado, y solo se permitirá pasar por el *firewall* aquellos que se permita explícitamente.

Como es obvio imaginar, la política por defecto ACEPTAR, facilita mucho la gestión del *firewall*, ya que simplemente nos tenemos que preocupar de proteger aquellos puertos o direcciones que sabemos que nos interesa, pero si no se protege explícitamente lo que nos importa, corremos riesgo en nuestra seguridad.

En cambio, si la política por defecto es DENEGAR, a no ser que lo permitamos explícitamente, el *firewall* se convierte en un auténtico muro infranqueable. El problema es que es mucho más difícil preparar un *firewall* así, y hay que tener muy claro cómo funciona el sistema implementando seguridad con *iptables* y que es lo que se tiene que abrir sin caer en la tentación de empezar a meter reglas super−permisivas y que la función del *firewall* sea ineficaz.

Iptables

Iptables es un sistema de *firewall* vinculado al *kernel* de Linux que se ha extendido enormemente. Los *iptables* están integrados con el *kernel* y son parte del sistema operativo.

Mediante *iptables*, se puede filtrar el tráfico por una gran variedad de criterios, mediante reglas, estás se integran en tres grupos diferenciados:

1. Reglas de entrada: para paquetes que llegan al *router*.
2. Reglas de encaminamiento: decisión sobre encaminar paquetes o no.
3. Reglas de salida: paquetes que salen del *router* por alguna interfaz.

Este programa es tan versátil que nos permite indicar en cada regla, la cadena (entrada, encaminamiento, salida), el **protocolo**, interfaz, direcciones de origen y destino, y el puerto utilizado en la conexión, así como la acción a tomar (denegar, rechazar, aceptar o aceptar y enmascarar) en caso de que determinado paquete cumpla la regla.

Para poner en marcha un *firewall* con *iptables*, se necesita únicamente aplicar reglas, para ello se ejecuta el comando *iptables*, con el que añadimos, borramos, o creamos reglas. Por lo tanto, un *firewall* de *iptables* no es sino un simple ***script*** de ***shell*** en el que se van ejecutando las reglas del *firewall*.

Al tener una máquina con Linux con soporte para *iptables*, se tienen reglas aplicadas y empiezan a llegar/salir/pasar paquetes. Las reglas de *firewall* están a nivel de *kernel*, y al *kernel* lo que le llega es un paquete y tiene que decidir qué hacer con él. El *kernel* lo que hace es, dependiendo si el paquete es para la propia máquina o para otra máquina, consulta las reglas de *firewall* y decide qué hacer con el paquete según mande el *firewall*.

En la figura 3.9 se muestra el camino que seguiría un paquete en el *kernel*:

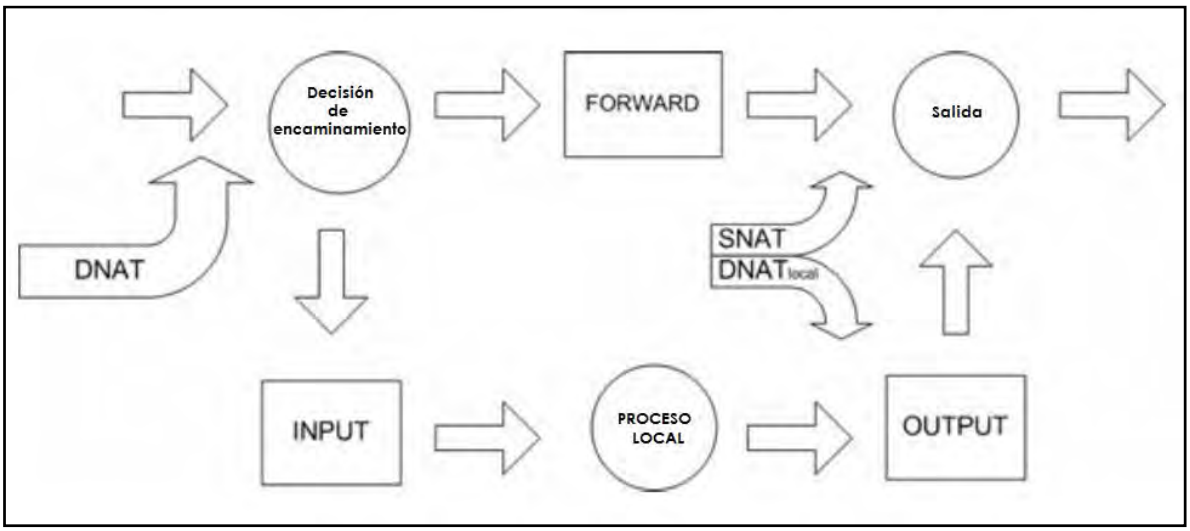


Figura 3.9 Trayecto de un paquete cuando llega al kernel con iptables.[55]

Como se ve en el gráfico, básicamente se observa si el paquete está destinado a la propia máquina o si va a otra.

Los *iptables* utilizan tres tipos de reglas:[56]

1. **NAT**, para realizar *Network Address Translation* formada por tres cadenas:

    I. ***Prerouting***, altera el paquete tan pronto como llega (antes de redireccionar).
    II. ***Output***, altera paquetes generados localmente.
    III. ***Postrouting*****,** altera el paquete antes de salir a la red.

2. MANGLE, modificación de paquetes y sus cabeceras, formada por dos cadenas:

    I. ***Prerouting***, altera el paquete tan pronto como llegue al firewall y antes de tomar la decisión de enrutar.
    II. ***Output***, alterar paquetes generados localmente antes de que se tome la decisión de enrutar.

3. FILTER, usadas generalmente para filtrar paquetes, por ejemplo DROP, LOG, ACCEPT o REJECT paquetes.

    I. ***Forward***, usado en todos los paquetes no generados localmente que no tienen como destino nuestro *host*.
    II. ***Input***, afecta a todos los paquetes con destino de nuestro host.
    III. ***Output***, afecta a los paquetes generados localmente.

Los *iptables* utilizan tres tablas diferentes para aplicar las reglas del *firewall*, las cuales son: para los paquetes (o datagramas, según el protocolo) que van a la propia máquina y se aplican las

[55] Altardill Izura, IPTABLES Manual práctico.
[56] Gozalbes Sanchis Vincent, Configuración de un firewall utilizando iptables.

reglas *INPUT* y *OUTPUT*, en cambio, para filtrar paquetes que van a otras redes o máquinas se aplican simplemente reglas *FORWARD.*

INPUT, OUTPUT y FORWARD son los tres tipos de reglas de filtrado. Pero antes de aplicar esas reglas es posible aplicar reglas de NAT, éstas se usan para hacer redirecciones de puertos o cambios en las ip's de origen y destino. Incluso antes de las reglas de NAT se pueden meter reglas de tipo **MANGLE**, destinadas a modificar los paquetes.

Con el preámbulo anterior podemos entender en el capítulo siguiente el script de Shell que se implementó para la seguridad del *cluster*.

3.3.3 Sincronización del tiempo

Todos los servidores usados con el *cluster* de replicación necesitan tener su reloj de tiempo real en sincronización, esto es para asegurar que el demonio slon no genere errores con mensajes indicando que un suscriptor está muy por delante de su proveedor durante la replicación.

Por lo anterior es recomendable tener el protocolo **NTP** corriendo en el nodo maestro para que a su vez los nodos suscriptores lo utilicen como su proveedor de tiempo.

NTP

El protocolo NTP (*Network Time Protocol*), es un protocolo de Internet ampliamente utilizado para transferir el tiempo a través de una red. NTP es normalmente utilizado para sincronizar el tiempo en clientes de red a una hora precisa.

Esquema de sincronización

Un servidor NTP primario o ***stratum****1*, está conectado a un reloj de referencia de alta precisión. Además, este servidor cuenta con *software* para manejar el protocolo NTP.

Otras computadoras, que funcionan como servidores *stratum2*, utilizan un *software* similar (usualmente el mismo), y consultan automáticamente al servidor primario para sincronizar su reloj. A su vez, éstos pueden sincronizar a otros servidores, que en este caso serán *stratum3*, y así podría seguirse hasta 16 niveles. La arquitectura también soporta que un cliente haga sus consultas a más de un servidor y puede haber comunicaciones entre servidores de un mismo *stratum*. En la figura 33 puede verse un esquema de esta estructura.

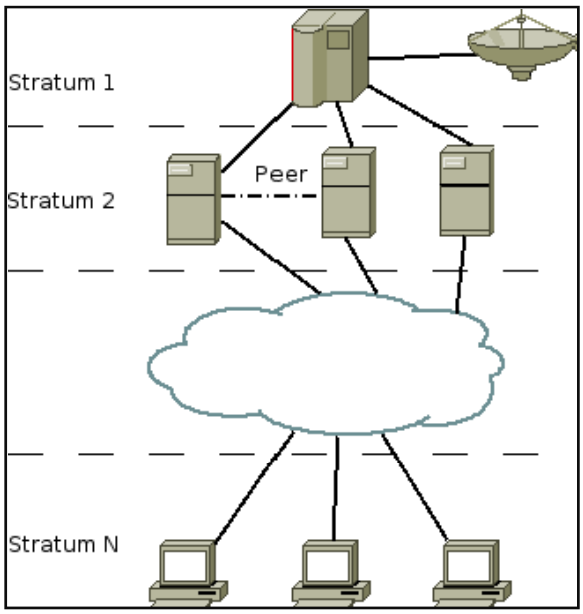


Figura 3.10 Esquema jerárquico de NTP.

Cuanto más alejado esté una computadora del reloj de referencia, o sea, cuanto más alto sea su *stratum*, menos precisa será la sincronización. Sin embargo, cualquier *stratum* siempre será suficiente para que el reloj no se aleje más de unos milisegundos de la hora real. Hasta ahora definimos que una máquina, que llamaremos cliente, puede sincronizarse con otra o con alguna referencia externa, y también puede comportarse como servidor, y utilizarse para sincronizar otras. Siempre que haya una asociación entre dos máquinas, donde una se comporte como cliente, y otra como servidor, al cliente le corresponderá el *stratum* inmediatamente superior al del servidor.

Para utilizar NTP en una organización, recomendamos instalar un servidor que se sincronice con varias fuentes externas. Este servidor será la única referencia horaria en la organización y todos los equipos estarán sincronizados con él.

El servidor NTP instalado servirá para que todos los equipos de la organización lo utilicen para ajustar sus relojes. Este ajuste será de gran importancia ya que permitirá, entre otras, la correlación de eventos entre diferentes equipos.

3.3.4 Respaldo

En la inmensa mayoría de los sistemas informáticos, los datos almacenados en el sistema tienen mucho mayor costo y son mucho más difíciles de recuperar que el sistema en sí. La pérdida de datos afecta la continuidad operativa de un sistema y principalmente ocurre por las siguientes causas:

- Sucesos incontrolados
    - Contingencias
    - Catástrofes naturales

- Fallas
    - En el *software*
    - En el *hardware*
    - Error humano

- Acciones controladas
    - En la realización de respaldos
    - En el mantenimiento y actualización de *hardware* y *software*

Es muy necesario observar todo aquello que tiene el potencial de dar un revés a nuestro sistema, pero más necesario aún el implementar un esquema de respaldo de todo lo se almacena en el sistema.

El respaldo de datos es la generación de una copia, en un momento determinado, de los datos del sistema, con el fin eventual de hacer una reposición de los datos en caso de pérdida. Algunas de las razones por las que se considera implementar un respaldo son las siguientes:

- Una de las razones fundamentales de contar con respaldos para contar con continuidad operativa, es decir, un sistema siempre disponible.
- La función de implementar estrategias de respaldo pretende minimizar los efectos negativos en el ámbito operacional por la interrupción de un servicio así como por la pérdida de información.

Para respaldar los datos del sistema, es necesario hacerlo en un medio que cumpla con una serie de exigencias, tales como:

Ser confiable: Minimizar las probabilidades de error, muchos medios magnéticos como las cintas de respaldo o discos duros tienen probabilidades de error o son particularmente sensibles a campos magnéticos, elementos que atentan contra la información que se ha respaldado.

Estar fuera de línea, en un lugar seguro: Tan pronto se realiza el respaldo de información, el soporte que almacena este respaldo debe ser desconectado de la computadora y almacenado en un lugar seguro tanto desde el punto de vista de sus requerimientos técnicos como humedad, temperatura, campos magnéticos, como de su seguridad física y lógica.

La forma de recuperación debe ser rápida y eficiente: Es necesario probar la confiabilidad del sistema de respaldo no sólo para respaldar sino que también para recuperar.

Los comandos tradicionales de respaldo y recuperación son *dump* y *restore*. El comando *dump* recorre el sistema de archivos haciendo una lista de los archivos modificados o nuevos desde una corrida anterior de *dump*; luego empaqueta todos esos archivos en uno solo.

Al crear un esquema de respaldo que se amolde a nuestras necesidades, también se deben tomar en cuenta las siguientes recomendaciones:

- Respaldar todo desde la misma máquina: aunque es más lento da la facilidad de administración y corrección del respaldo.

- Etiquetar los respaldos físicos: fecha, hora, máquina, sistema de archivos, número serial constituyen el mínimo absoluto. Un registro más completo se hace imprescindible cuando se respaldan muchos sistemas de archivos en un mismo volumen.

- Intervalo de respaldos razonable: el sistema de archivos de usuarios puede respaldarse a diario en sistemas grandes, o semanalmente en sistemas chicos, la frecuencia de respaldo para otros sistemas de archivos dependerán del uso y la criticidad.

- Respaldos diarios en un solo volumen: buscar un esquema o un medio para que el respaldo diario quepa en un solo volumen; es la única forma simple de realizar un respaldo diario.

- Seguridad del respaldo: el robo de un respaldo equivale al robo de toda la información vital de una organización. Las precauciones de seguridad con los respaldos debe ser tanta como la implementada para el propio sistema.

- Limitar la actividad durante *dump*: la actividad en los archivos mientras se ejecuta *dump* puede ocasionar confusión en el momento de restaurar. Puede hacerse el respaldo en horas de escasa actividad, nocturnas o en fin de semana. Es preciso cuidar de no coincidir con los programas del sistema que modifican el sistema de archivos; éste debe permanecer estacionario mientras se realiza el respaldo.

Con las recomendaciones anteriores sugeridas es necesario crear un esquema de respaldo que ajuste a las necesidades de la geDGAPA con la finalidad de respaldar toda la información de carácter crítico y evitar que algún factor afecte la continuidad operativa de los sistemas manejados por la geDGAPA.

3.3.4 Migración

Para el proceso de migración se utilizaran dos herramientas de gran utilidad para comparar la estructura y los datos de las bases de datos a migrar, las cuales son:

I. Data Comparer para PostgreSQL
II. DB Comparer para PostgreSQL

Data Comparer para PostgreSQL

Data Comparer para PostgreSQL es una herramienta poderosa y fácil de usar para la comparación y sincronización de datos.

Se utiliza para automatizar las migraciones de datos, análisis de datos dañados, restablecer respaldos de datos y para agregar datos faltantes o que fueron cambiados. Es la herramienta estándar de la industria para comparar y sincronizar el contenido de dos bases de datos porque es muy fiable y también muy rápido.

De manera más específica las acciones que se pueden realizar con Data Comparer son las siguientes:

- Comparar el contenido de dos bases de datos
- Sincronizar automáticamente sus datos
- Trabajar con secuencias de comandos SQL, *backups* o bases en línea.

Data Comparer ahorra muchas horas ya que evita que se pierda el tiempo copiando las bases de datos, restaurándolas y comparándolas con bases de datos de respaldo a mano, además, se puede especificar exactamente qué tablas, filas y columnas de las bases de datos que desean sincronizar.

Algunas de las cosas que Data Comparer hace por nosotros son:

- Verificar el éxito de la migración de datos.
- Restaurar los datos que faltan o se encuentren dañados en una tabla, fila o columna.
- Solucionar problemas de replicación.
- Migrar los datos sin problemas entre PostgreSQL 7 y 8.
- Almacenar datos estáticos en carpetas para facilitar el control de los datos.
- Proporcionar una herramienta de control de calidad y documentación
- Mostrar los datos en columnas con un pivote para acelerar el análisis de diferencias.
- Ejecutar respaldos automáticos antes de comenzar la sincronización.
- Opcionalmente ofrece la compresión de archivos temporales para mejorar el rendimiento.

A continuación se muestra la ventana de configuración del Data Comparer, donde se colocan los datos de los servidores que se desean comparar (Figura 3.11).

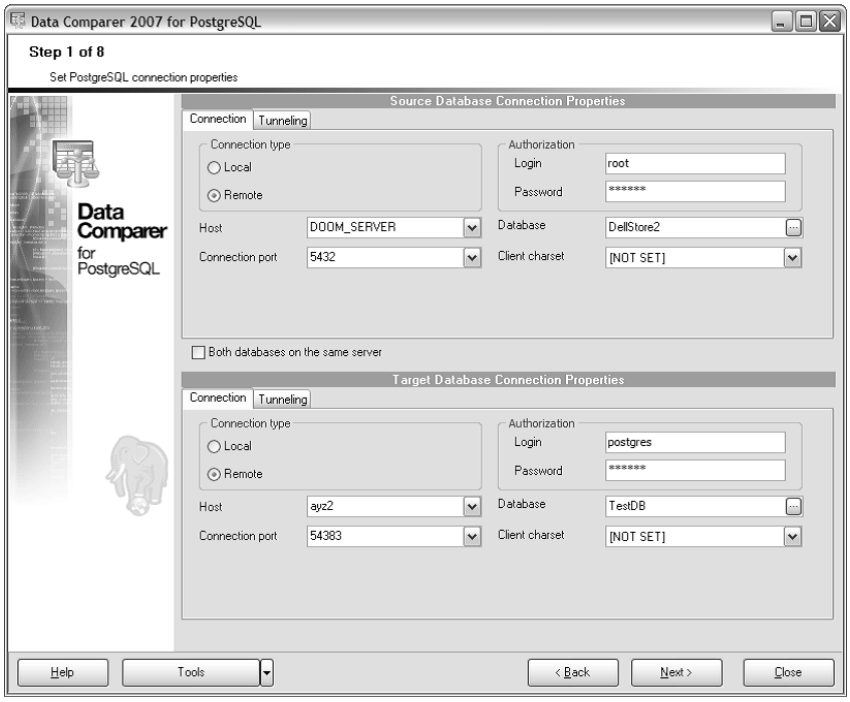


Figura 3.11 Ventana de configuración en el DataComparer.[57]

[57] DataComparer 2007 for PostgreSQL User's manual.

Por lo tanto, esta herramienta se convierte en un componente importante dentro del *cluster* y específicamente dentro la etapa de migración.

DB Comparer para PostgreSQL

DB Comparer para PostgreSQL es una excelente herramienta para la comparación de bases de datos para PostgreSQL y la sincronización.

Permite ver todas las diferencias en la comparación de bases de datos de objetos y ejecutar un *script* que genere automáticamente la sincronización de la estructura de las bases de datos a migrar y eliminar todas las diferencias que puedan existir.

El programa tiene la capacidad de automatizar los esquemas de base de datos, comparar y sincronizar tareas usando la consola de la herramienta. Su interfaz fácil de usar simplifica enormemente el descubrimiento y la eliminación de las diferencias en la estructura de base de datos PostgreSQL ahorrando bastante tiempo.

Algunas de las características principales que posee DB Compare*r* son:

- Compara y sincroniza esquemas de bases de datos en diferentes servidores, así como se puede realizar en un solo servidor.
- Compara todos los objetos de las bases de datos o sólo los seleccionados.
- Representación visual de las diferencias entre las bases de datos con los detalles y la modificación de los *scripts* de los diferentes objetos.
- Capacidad para sincronizar bases de datos manualmente, paso a paso o automáticamente.
- Capacidad para generar informes con las diferencias entre las bases de datos.
- Capacidad de automatizar la comparación y sincronización de las bases de datos utilizando la consola.
- Se pueden comparar varios proyectos a la vez.
- Guardar y cargas proyectos con todos sus parámetros.
- Una gran variedad de opciones para realizar la comparación y sincronización de las bases de datos.

A continuación se muestra la ventana principal de DB Comparer en la figura 3.12.

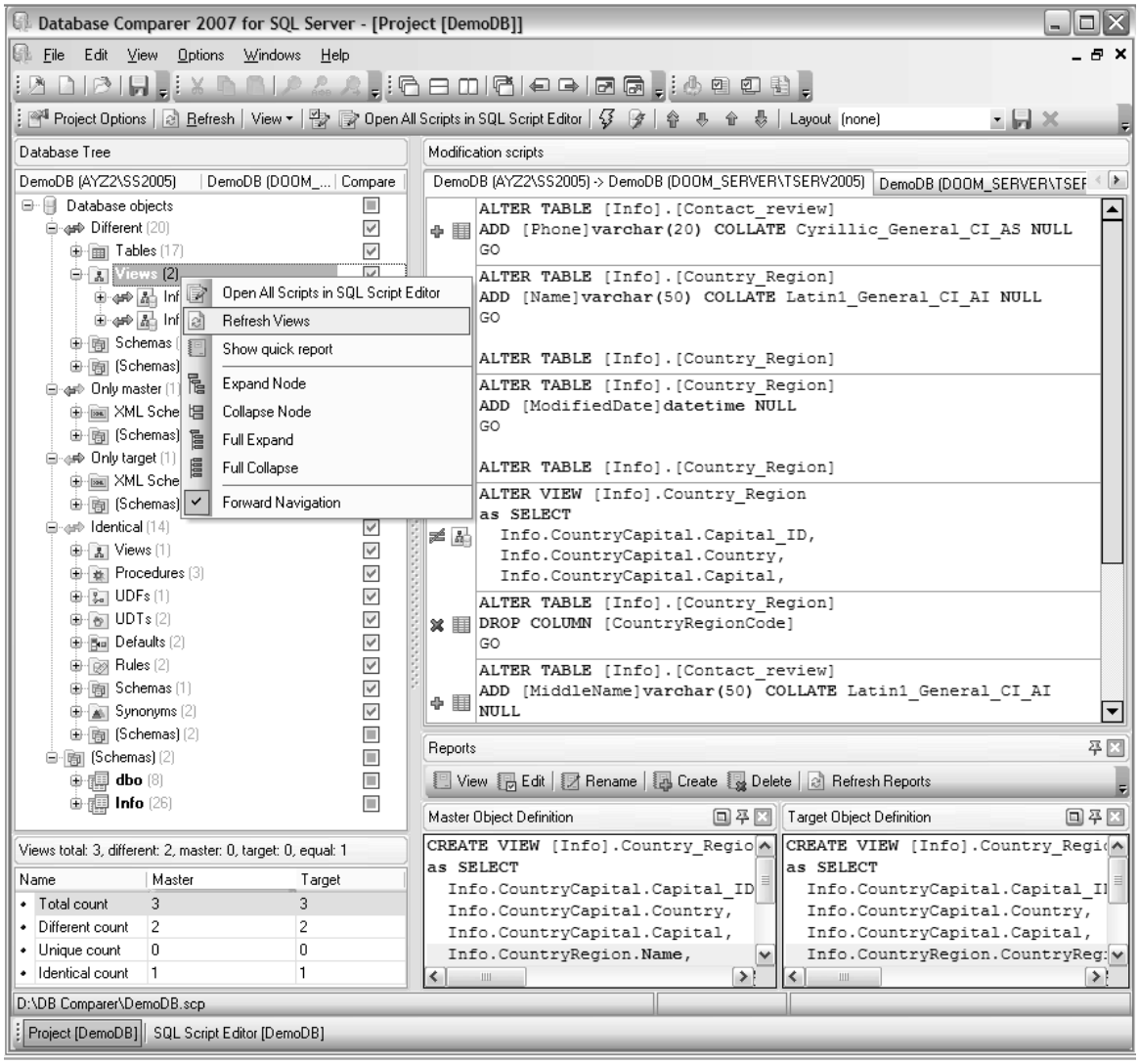


Figura 3.12 Ventana principal de DBComparer.[58]

Con la descripción general de las dos herramientas ocupadas para la migración, Data Comparer y DB Comparer para PostgreSQL, donde la primera se encarga de analizar los datos de las bases y la segunda realiza la comparación en la estructura de la bases de datos, para que juntas sean el complemento perfecto para una correcta migración de los datos.

3.3.5 Consola de administración del cluster

Un componente importante del *cluster* son las bases de datos que se manejan dentro de la replicación, por ello, resulta conveniente tener una herramienta que nos proporcione administrar dichas bases de datos, de cada uno de los nodos que componen el *cluster,* con la finalidad de hacer más accesible consultar, actualizar, borrar, entre otras más, es decir, realizar operaciones ordinarias en las bases de datos, mediante una interfaz gráfica fácil de utilizar.

[58] DB Comparer 2007 for PostgreSQL User's manual

Para el *cluster* la herramienta elegida fue el *software* gratuito de phpPgAdmin-4.2.2, a continuación se detallarán las características del mismo.

phpPgAdmin

phpPgAdmin es una aplicación web, desarrollada en **PHP**, para administrar bases de datos de PostgreSQL.

phpPgAdmin provee una manera conveniente a los usuarios para crear bases de datos, tablas, alterarlas y consultar sus datos usando el lenguaje estándar **SQL**.
phpPgAdmin estuvo basado en phpMyAdmin, pero hoy día ya no comparte código con él; incluso provee las mismas funcionalidades y más a los usuarios del servidor de base de datos PostgreSQL.

Algunas de las características más relevantes de phpPgAdmin son las siguientes[59]:

- Administrar múltiples servidores.
- Soporte para PostgreSQL 7.0.x, 7.1.xy, 7.2.x, 7.3.x, 7.4.x, 8.0.x, 8.1.x, 8.2.x, 8.3.x
- Gestionar todos los aspectos de:
    - Usuarios y grupos
    - Bases de datos
    - Esquemas
    - Tablas, índices, ***constraints***, ***triggers***, reglas y privilegios
    - Vistas, funciones y secuencias
    - Informes
- Fácil manipulación de datos:
    - Exploración de tablas, vistas e informes
    - Ejecución de sentencias SQL
    - Operaciones de selección, inserción, actualización y borrado
- Volcado de los datos de las tablas en una variedad de formatos: SQL, XML, XHTML, CSV, con un *pg_dump*
- Importar scripts SQL
- Soporta el motor de replicación maestro – esclavo de Slony
- Excelente soporte de idiomas:
    - Disponible en 27 idiomas
    - No presenta conflictos de codificación
- Fácil de instalar y configurar

A continuación en las figuras 3.13 y 3.14 se muestran algunas vistas de la consola de administración de phpPgAdmin:

[59] http://phppgadmin.sourceforge.net/
phpPgAdmin.

Figura 3.13 Vista de una base de datos.[60]

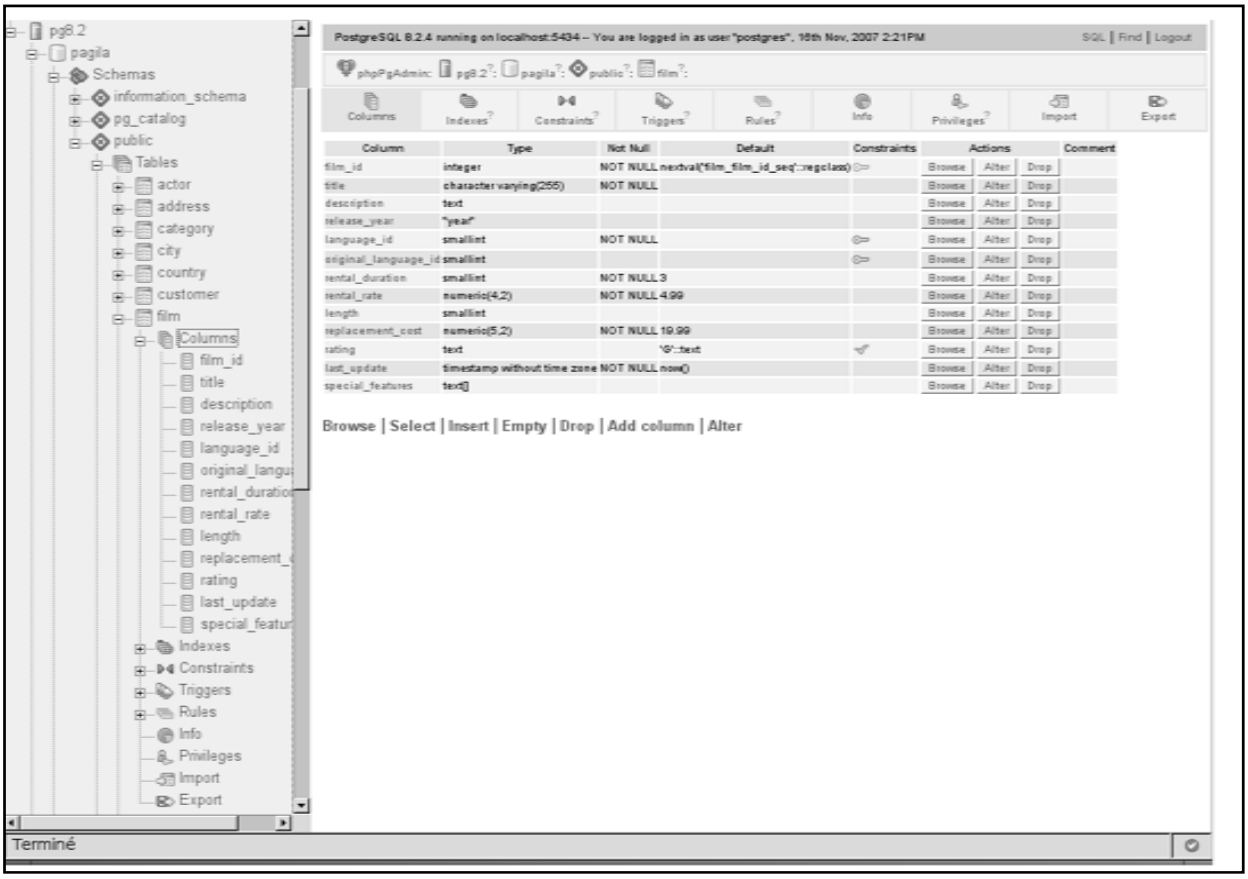


Figura 3.14 Vista de la estructura de una tabla.[61]

[60] http://phppgadmin.sourceforge.net/
[61] Idem.

3.3.6 Pgpool-II

Además de Slony-I como *software* de replicación, se contará con un *software* adicional que permitirá el proceso de *failover*, el cual es Pgpool-II, a este componente del *cluster* lo denominaremos *failover-switch.*

Pgpool-II es un *middleware* o en su traducción, un *software* intermediario que trabaja entre los servidores de PostgreSQL y el cliente de las bases de datos de PostgreSQL.

Pgpool-II habla los protocolos de ***frontend*** y ***backend*** de PostgreSQL, y pasa las conexiones entre ellos. De ese modo, una aplicación de base de datos (*frontend*) cree que Pgpool-II es el verdadero servidor de PostgreSQL, y el servidor (*backend*) ve a Pgpool-II como uno de sus clientes. Pgpool-II es una herramienta transparente tanto del lado del cliente como del lado del servidor.

Pgpool-II funciona sobre Linux, Solaris, FreeBSD y la mayoría de las arquitecturas UNIX. Las versiones de PostgreSQL soportadas son de la 6.4 para arriba. Para usar la paralelización de consultas es necesaria la versión 7.4 o superior.

Pgpool-II proporciona las siguientes características según su configuración:

- Limita el excedente de conexiones

  PostgreSQL soporta un cierto número de conexiones concurrentes y rechaza las que superen dicha cifra. Aumentar el límite máximo de conexiones incrementa el consumo de recursos y afecta al rendimiento del sistema. Pgpool-II tiene también un límite máximo de conexiones, pero las conexiones extras se mantienen en una cola en lugar de devolver un error inmediatamente.

- Pool de conexiones

  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.

- Replicación

  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.

- Balanceo de carga

  Si se replica una base de datos, la ejecución de una consulta **SELECT** en cualquiera de los servidores devolverá el mismo resultado. Pgpool-II se aprovecha de 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 cuál hay muchos usuarios ejecutando muchas consultas al mismo tiempo.

- Paralelización de consultas

  Al usar la función de paralelización de consultas, los datos 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.

Además Pgpool-II brinda distintas posibilidad de configuración según el objetivo deseado, es decir, puede configurarse por ejemplo, como balanceador de cargas, de *failover,* de modo paralelo entre otras, como se muestra en la tabla 3.2.[62]

Tabla 3.2 Modos de configuración de Pgpool-II

Modo / *Función*	*Modo crudo (Raw Mode)*	*Modo de pool de conexión (Connection Pool Mode)*	*Modo de replicación (Replication Mode)*	*Modo Maestro/Esclavo (Master/Slave Mode)*	*Modo de query paralelo (Parallel Query Mode)*
Pool de Conexión	*X*	*O*	*O*	*O*	*O*
Replicación	*X*	*X*	*O*	*X*	*(*)*
Balanceo de carga	*X*	*X*	*O*	*O*	*(*)*
Failover	*O*	*O*	*X*	*O*	*X*
Query paralelo	*X*	*X*	*X*	*X*	*O*
N° de servidores mínimo	*1 ó más*	*1 ó más*	*2 ó más*	*2 ó más*	*2 ó más*

Donde:

X: No cuenta con el modo

0: Cuenta con el modo

(*): La función de replicación y el balanceo de cargas no pueden ser usados por la tabla que presenta información dividida por el modo de *query* paralelo.

Por lo tanto, Pgpool-II se convierte en una herramienta de utilidad, específicamente para el proceso de *failover*, Pgpool-II entrará cuando un nodo se convierta en no disponible y sea necesario hacer el cambio del mismo dentro del sistema de replicación.

Al contar con Slony-I como replicador se puede hablar de alta disponibilidad, porque además del nodo maestro como el origen de datos, se cuenta con dos nodos esclavos que contienen la misma información como una copia total del nodo maestro, pero esto no es suficiente, porque cuando un nodo falle no hay una herramienta que haga el cambio entre un

[62] http://pgpool.projects.postgresql.org/pgpool-II/doc/pgpool-en.html
Pgppool user´s manual

nodo y otro, se tiene la copia pero no quien realice dicho cambio, entonces, es donde Pgpool-II será configurado para realizar dicho cambio como un proceso de *failover*.

Pgpool-II se encargará de hacer el cambio de un nodo caído por otro que funcione correctamente, entonces ahora si se puede hablar de un sistema completo de alta disponibilidad, por que se tienen los datos replicados y una herramienta para manejar los mismos, en caso de una contingencia o de una tarea de mantenimiento.

A continuación en la figura 3.15 se muestra el menú de opciones que brinda la consola de administración de pgPoolAdmin:

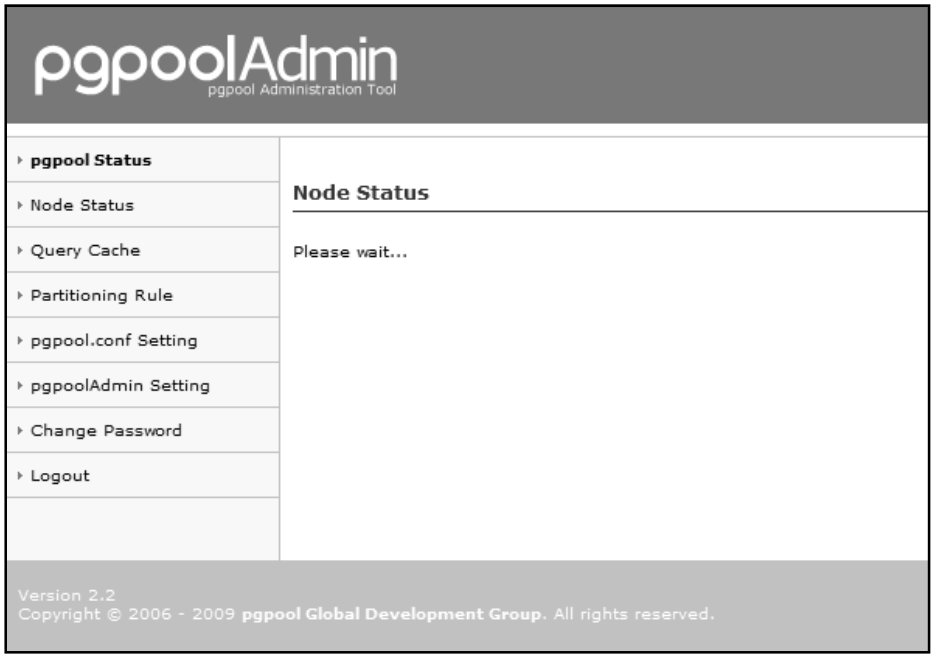


Figura 3.15 Consola de administración de Pgpool-II.

3.3.7 Monitoreo

El Monitoreo es el proceso continuo y sistemático mediante el cual verificamos la eficiencia de un sistema, mediante la identificación de sus logros y debilidades y en consecuencia, se diseñan medidas correctivas para optimizar los resultados esperados.

Los problemas que ocurren con cierta frecuencia dentro de un sistema con red local, son los siguientes:

- Caída de servidores
- Denegación de servicio
- Servicios dados de baja
- Demanda que supera la capacidad de transmisión.

Ante la problemática anterior surgen las siguientes necesidades:

- Contar con información oportuna para dar soluciones.
- Medir la eficacia de la red utilizada para prevenir errores futuros que pueda presentar.
- Tener en cuenta la tasa de transferencia para identificar la carga de la red.
- Llevar un registro histórico de la información.
- Disminuir costos.
- Proveer una solución multiplataforma.

Para los problemas mencionados anteriormente, es necesario plantear una solución para tener un plan de contingencia ante ellos.

Para el *cluster* es necesario implementar distintos servicios de monitoreo, enfocados cada uno a los componentes que se consideren que poseen funciones críticas para el correcto desempeño del *cluster*.

Dentro del diseño del *cluster* se contemplan dos herramientas de monitoreo con distintas características pero una misma finalidad, optimizar el *cluster,* en base al análisis de los resultados de dichos monitores.

Los tres monitores a ocupar en el *cluster* son:

1. Mon
2. Smart Mon Tools
3. Power Alert

1. Mon

Mon es una herramienta que fue desarrollada para monitorear la disponibilidad de los servicios y tiene la capacidad de mandar alertas sobre eventos previamente definidos en una red.

Una de las metas del diseño de Mon es mantener la simplicidad y proveer correctamente el monitoreo, para que el sistema pueda ser escalable, fácil de usar y que se pueda extender a una gran variedad de aplicaciones.[63]

Mon es un *software* ***Open Source*** y es distribuido bajo la licencia GNU y sus características principales son:

- Es portable (desarrollado en **Perl**).
- Tiene un diseño adaptable.
- Puede monitorear cualquier dispositivo en la red.
- Configurable con otros sistemas.
- Es fácil de implementar alertas y monitores.

El diagrama de la estructura funcional de Mon es la que se muestra en la figura 3.16.

[63] http://mon.wiki.kernel.org
Mon – Service Monitoring Daemon.

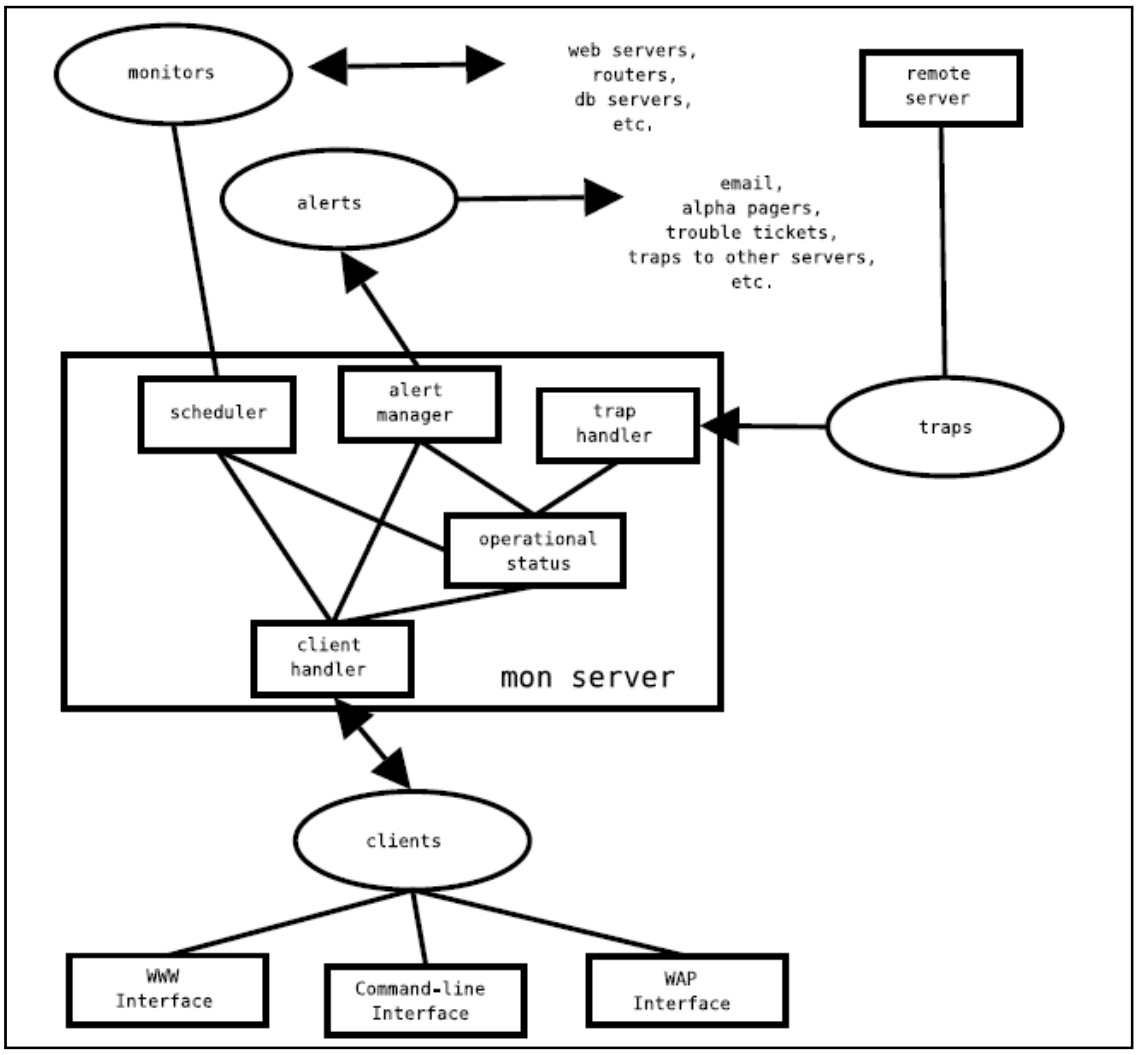


Figura 3.16 Diagrama funcional de MON.[64]

Como se observa en el diagrama, se encuentran los clientes que son los servidores que se desean monitorear, más arriba se observa los componentes del motor de Mon, que a su vez funciona de manera remota, es decir, en un servidor externo (*remote server*) y desde ahí tiene monitoreado los servicios de los clientes, dentro del motor de Mon se encuentra el administrador de alertas (*alert manager*) que tiene la función de mandar una alerta vía mail a los administradores de la red, cuando algún servicio de un cliente se encuentre fuera de servicio.

Al utilizar Mon como *software* de monitoreo se obtienen las siguientes ventajas:

- Tener un monitoreo de cada uno de los servicios dentro de cada uno de los servidores dentro de una red, puede monitorear casi todo, sin clientes ni agentes.
- Mon tiene la característica de agregar de manera simple alertas y monitores con el fin de alertar a los administradores de la red cuando un servidor se encuentre abajo o fuera de servicio.

[64] http://mon.wiki.kernel.org
Mon – Service Monitoring Daemon

- Configurable, expandible y puede integrarse con otros sistemas.
- Es fácil hacer pruebas y alertas en las conexiones.
- Manera simple de recopilar los datos para la generación de informes.
- De propósito general, si se puede revisar el servicio, se puede monitorear.

En la figura 3.17 se observa la consola de Mon, donde se muestra el estado en que se encuentra los servicios monitoreados por la consola.

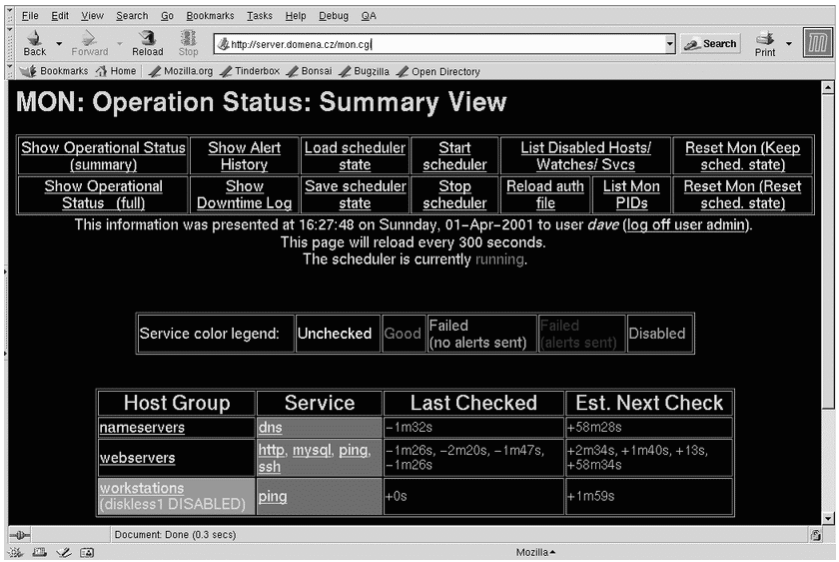


Figura 3.17 Ejemplo de la consola de MON.[65]

Por lo tanto, el monitoreo de los servicios es una condición para la rectificación o profundización de la ejecución de un sistema.

2. Smart Mon Tools

Tecnología S.M.A.R.T.

Todos los discos duros que se usan hoy en día poseen la tecnología denominada **S.M.A.R.T.** (*Self Monitoring Analysis and Reporting Technology*), ésta tecnología permite detectar por anticipado posibles errores físicos del disco duro, como[66]:

- la velocidad de los platos del disco,
- sectores defectuosos
- errores de calibración
- temperatura del disco

[65] http://mon.wiki.kernel.org
Mon – Service Monitoring Daemon.

[66] http://jorge.huerga.org/2008/07/prevenir-errores-de-disco-duro-gracias-a-smart/
Tecnología SMART por Jorge Huerga.

Con la finalidad de prevenir fallos en el disco duro antes que sea demasiado tarde y el disco duro quede totalmente inutilizado.

Existen dos tipos de categorías de errores, los impredecibles (fallos de voltaje, temperaturas elevadas, errores en algún circuito integrado…) y los predecibles que son fallos mecánicos que se van desarrollando paulatinamente a medida que el uso del disco va siendo mayor (también depende de la calidad del dispositivo).

Para poder tener tranquilidad con nuestros discos duros, sobre todo si son usados 24 horas al día, podemos monitorizar nuestros discos duros usando esta tecnología. Los parámetros que hay que tener controlados para monitorizar el disco son los siguientes:

- Temperatura del disco: Un aumento de temperatura excesivo del disco puede hacer que funcione mal los elementos electrónicos.
- Tasa de transferencia: Si van aumentando los errores en la tasa de transferencia es síntoma de que puede haber algún error.
- Velocidad de lectura: Muy similar al punto anterior, y la tasa de error puede provenir por síntomas muy similares.
- Tiempo de partida *(spin-up)*: Puede ser un reflejo de algún tipo de error de motor del disco.
- Contador de sectores reasignados: Cuando son reasignados muchos valores, significa que no pueden ser grabados donde se intenta hacerlo, con lo cual deben ser asignados a otros sectores. Esto es una referencia inequívoca del deterioro del disco y de su fallo probablemente inmediato.
- Altura de Vuelo del Cabezal: La tendencia a la baja en altura de vuelo a menudo presagian un accidente del cabezal, esto significa que puede haber uno de los peores errores de un disco duro. Que el cabezal dañe la superficie física del disco duro y lo deje totalmente inutilizable.
- Uso de **ECC** y Conteo de errores: Es un dato importante a tener en cuenta, el número de errores detectados por la unidad, aunque se corrijan internamente, a menudo señala problemas con su desarrollo. La tendencia es, en algunos casos, más importante que el conteo real.

Los valores de los atributos S.M.A.R.T. van del número 1 al 253, siendo 1 el peor valor. Los valores normales son entre 100 y 200. Estos valores son guardados en un espacio reservado del disco duro.

Solución para Linux: Smart Mon Tools.

Antes de explicar en qué consiste Smart Mon Tools, es necesario saber que existe un sistema que permite mantener dos discos a la vez siendo uno espejo del otro, de modo que si uno tiene un fallo físico, siempre queda el otro disco. Éste sistema se denomina **R.A.I.D.** *(Redundant Array of Inexpensive Disks*), que aunque como su nombre indica es un sistema redundante "barato" de conjunto de discos, sigue siendo el doble de caro que tener un solo disco.

Para ahorrar energía eléctrica y no gastar dinero en tener dos discos, podemos instalar en nuestro sistema la solución Smart Mon Tools una combinación de ambas tecnologías, R.A.I.D. y Smart Mon Tools conseguirían una tolerancia a fallos más que satisfactoria.

Smart Mon Tools soporta discos **SATA** y discos **SCSI** y dispositivos de cinta.

Smart Mon Tools contiene dos útiles programas[67]:

1. Smartcrl
2. Smartd

Smartd

Smartd es un demonio que monitorea el análisis de la tecnología S.M.A.R.T., esté se enfoca a monitorear los dispositivos ATA y SCSI cada cierto tiempo, configurable según se desee.

También se pueden mandar alertas vía e-mail si un problema es detectado, además será conveniente hacer un auto chequeo del disco que presenta problemas, respaldar el disco, reemplazarlo o utilizar una herramienta que reubique los sectores dañados de un disco.

Smartcrl

Es una utilidad de línea de comandos diseñada para realizar tareas enfocadas a la tecnología S.M.A.R.T, como imprimir los *logs* de error, habilitar y deshabilitar pruebas automáticas de S.M.A.R.T, el usuario debe especificar el dispositivo que se desea monitorear por smartcrl.

Con dichos programas Smart Mon Tools brinda la posibilidad de monitorear, analizar y alertar sobre el estado de los discos físicos de los equipos para prevenir fallas en los mismos.

3. Power Alert

UPS

Un **UPS** (*Uninterruptible Power Supply)* es un sistema completo de respaldo de energía eléctrica capaz de mantener operando el equipo conectado a él, por un periodo de tiempo mientras no haya suministro eléctrico, con la finalidad de no perder la operación del equipo. Usualmente los UPS tienen sistemas sofisticados de protección como son supresores de picos, filtros, reguladores de voltaje, etc.[68]

Otra de las funciones del UPS es la de mejorar la calidad de la energía eléctrica que llega a los aparatos, filtrando subidas y bajadas de tensión y eliminando armónicos de la red en el caso de corriente alterna. Los UPS dan energía eléctrica a equipos de cargas críticas, que pueden ser aparatos médicos, industriales o informáticos, que requieren tener siempre alimentación y que ésta sea de calidad debido a la necesidad de estar en todo momento operativos y sin fallos (picos o caídas de tensión).

[67] http://smartmontools.sourceforge.net/
Smart Mon Tools Home Page

[68] http://www.maestrosdelweb.com/principiantes/cuidar/
¿Cómo alargar la vida de nuestro equipo de computación? por Christian Van Der Hents.

Power Alert

Power Alert es un sistema de administración de energía en una red, permite al administrador de red para supervisar y controlar hasta 250 UPS desde una sola interface, es conveniente mencionar que este *software* es gratuito pero solo funciona en los UPS de la marca TrippLite.

Usando **Java** y el protocolo **SNMP**, Power Alert simplifica la administración de energía desde un simple servidor hasta una red corporativa.

Power Alert permite a los administradores de red un seguimiento centralizado de todos los UPS dentro de la red, además permite configurar los parámetros en los que los UPS serán apagados automáticamente en el caso de que ocurra un fallo eléctrico que pudiera extenderse por más del tiempo que un UPS pueda brindar al equipo. [69]

A continuación en la figura 3.18 se muestra el diagrama del funcionamiento de Power Alert, se tienen dos UPS conectados a un servidor cada uno, y a su vez se encuentran conectados a la red mediante TCP/IP, a la misma red se conectará un equipo que se encargará de monitorear los servidores mediante la consola de administración de Power Alert.

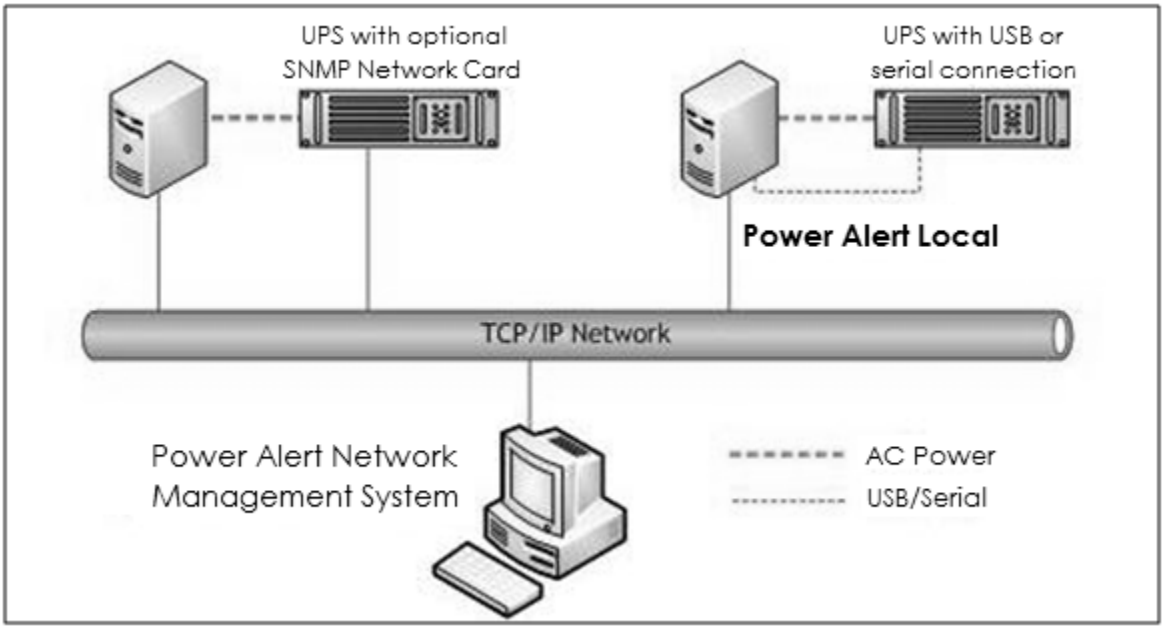


Figura 3.18 Diagrama del funcionamiento de Power Alert.[70]

También a continuación se muestra en las figuras 3.19 y 3.20, vistas de la consola de administración de Power Alert.

[69] http://www.tripplite.com/en/support/PowerAlert/index.cfm
Tripp-Lite Support.
[70] Idem.

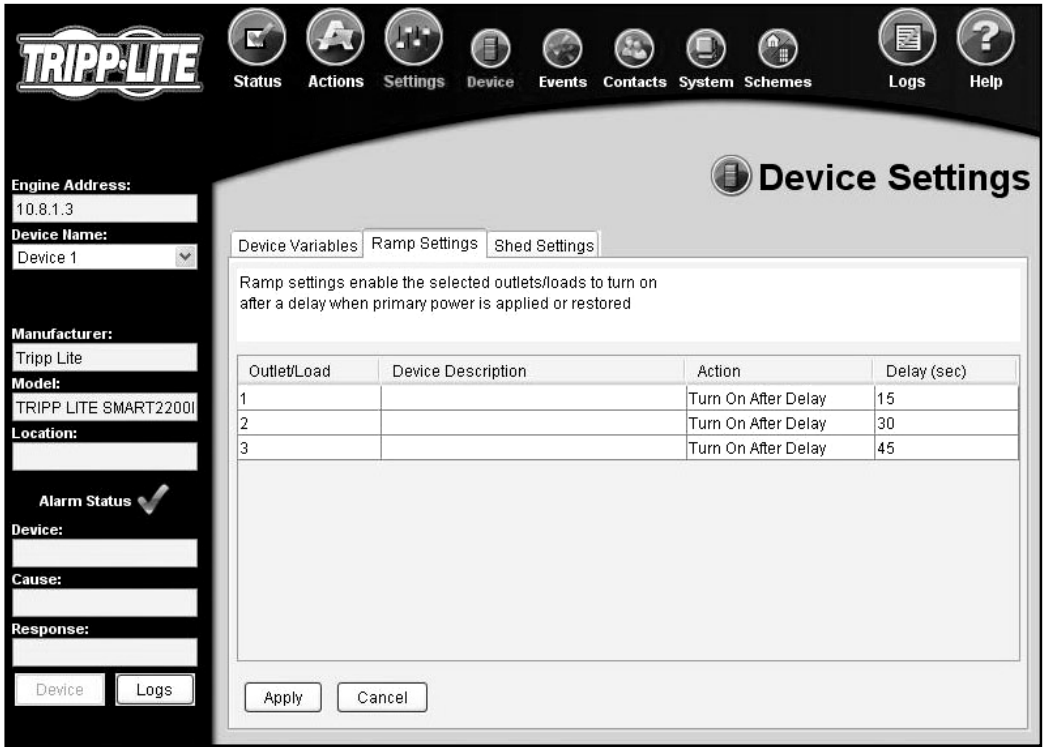


Figura 3.19 Características del UPS.[71]

Figura 3.20 Ventana del estado de los dispositivos UPS.[72]

[71] http://www.tripplite.com/en/support/pdf/PowerAlertUserManual045.pdf
User's Guide Power Alert Software Version 12.04.0045

[72] Idem.

Por lo tanto, el uso de estos programas sirve para controlar y supervisar los sistemas de almacenamiento usando el auto-monitoreo, análisis y presentación de informes de la tecnología S.M.A.R.T incorporada en la mayoría de los discos duros ATA y SCSI. En muchos casos estos programas proporcionan una forma avanzada de alertar de la degradación del disco y su fallo.

3.3.8 *Hardware* del cluster

En este subcapítulo, se dará la descripción de los equipos físicos que conformarán el *cluster*, un análisis detallado del *hardware* a utilizar en la implementación del *cluster*.

La dependencia tiene disponible para poner en funcionamiento el *cluster* equipos con la siguiente tecnología, mostrados en la tabla 3.2:

Tabla 3.2 Características del servidor Sun Fire X4200

Modelo	Sun Fire X4200
Microprocesador	2x AMD Opteron a 2.8Ghz/1Mb
Memoria RAM	4x 2Gb
Disco Duro	2x 73Gb
Tarjeta de red	4x 10/100/1000 Ethernet

Los equipos que conforman el *cluster* son de gran importancia porque en ellos recaen el funcionamiento adecuado del *software* que se instale en ellos, si no se cuenta con los equipos adecuados el funcionamiento del cluster no será el deseado, por lo tanto, el *hardware* es la infraestructura *a priori* del *cluster*, por ello, resulta necesario conocer la tecnología con la que se cuenta y aprovechar dichos recursos de la manera más conveniente.

3.3.9 *Software* del cluster

Ahora se especificará el *software* a utilizar por los equipos mencionados anteriormente y así como el *hardware* con el que se dispone es importante, el *software* a utilizar en dicho *hardware* es el complemento para sacar el mayor aprovechamiento a los recursos con los que se cuenta.

El *software* a utilizar en la implementación del *cluster* en su totalidad es de licencia libre, esto quiere decir que está al alcance de cualquier persona que se interese en el enfoque que propone este trabajo de tesis.

En la tabla 3.3 se resume el *software* que compone el funcionamiento del *cluster*.

Tabla 3.3 *Software* de los componentes del cluster.

COMPONENTE	VERSIÓN
Sistema operativo:	Ubuntu server 8
Manejador de base de datos:	PostgreSQL 8.2.4
Replicador de datos:	Slony-I 1.2.15
Protocolo de sincronización de reloj:	NTP
Failover switch:	Pgpool II
Consola de Administración:	phpPgAdmin
Herramienta de Migración:	DataComparer para PostgreSQL DB Comparer para PostgreSQL
Monitores del cluster:	MON Power Alert Smart MonTools

3.4 Diseño

3.4.1 Diseño físico del cluster

A continuación se muestra en la figura 3.21 el arreglo de los componentes del *cluster*:

1. En primera instancia se tiene el servidor web donde acceden desde internet los usuarios de la DGAPA, cuando necesitan realizar un trámite en línea de las distintas convocatorias que publica la dependencia.

2. Después del servidor web se encuentra el servidor de aplicaciones, que es el lugar donde se aloja la programación de las distintas aplicaciones a las que los usuarios piden acceso.

3. Posteriormente se encuentra el *failover-switch* que se encargará de establecer la comunicación con el nodo maestro de PostgreSQL, con la ayuda de Pgpool se realizara el *switcheo* de nodo maestro cuando así se requiera, como en los casos de mantenimiento del nodo o falla del mismo.

4. Finalmente se encuentra el nodo maestro y los dos nodos esclavos que mediante Slony-I se mantendrán en comunicación, donde el nodo maestro se encargará de replicar los datos existentes y nuevos a los nodos restantes para así cumplir con el propósito de alta disponibilidad del sistema.

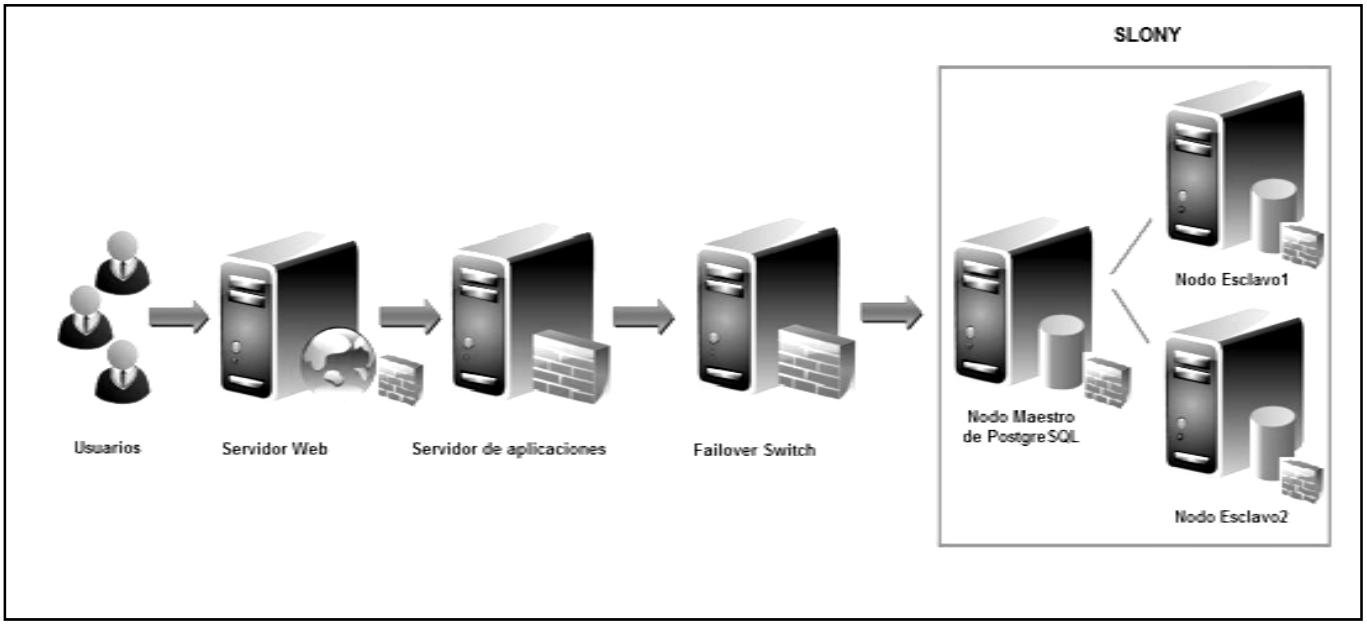


Figura 3.21 Diseño del cluster.

3.4.2 Diseño de los sets de replicación de Slony-I

En subcapítulos anteriores, se explicó el funcionamiento del motor de Slony, el cual mencionaba que para realizar la replicación de los datos, se creaban sets de replicación que contenían los datos a replicarse entre los nodos.

A continuación en la figura 3.22 se muestra el escenario de replicación de los sets para el cluster:

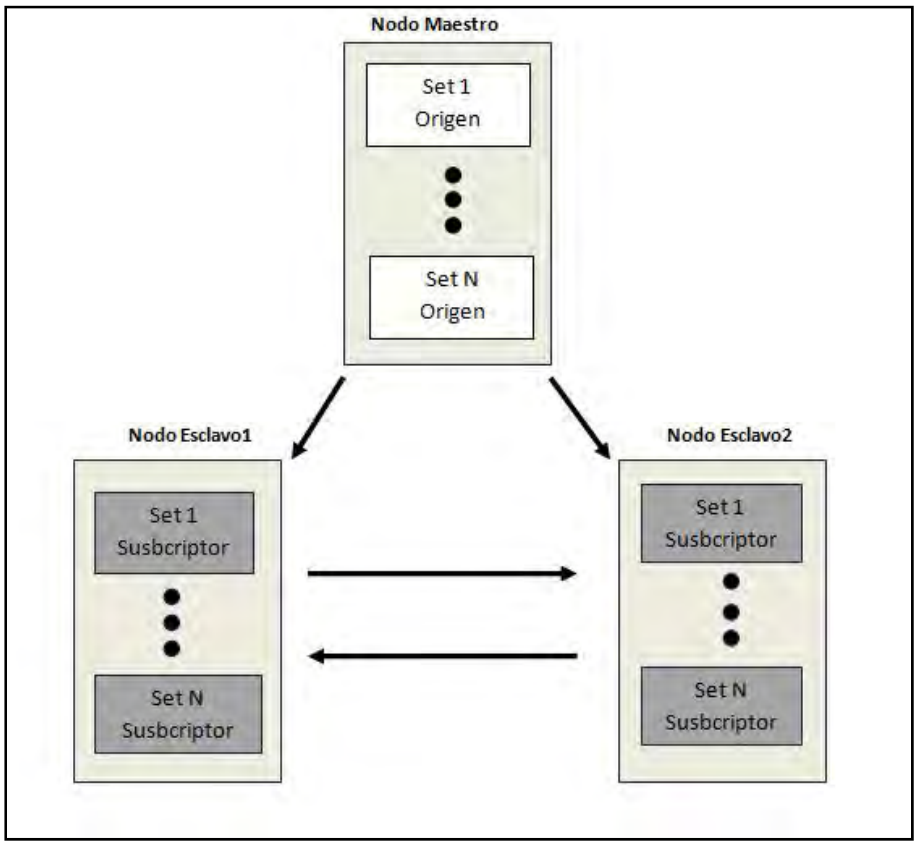


Figura 3.22 Replicación de los sets en el cluster.

Se tiene un escenario donde el nodo maestro es configurado para replicar desde el *set* 1 hasta el *set* N, tanto en el nodo esclavo 1 como en el nodo esclavo 2.

El evento *SYNC* generado en el nodo maestro es mandado a ambos nodos para que actualicen los datos replicados y a su vez el nodo esclavo 1 y esclavo 2 mantienen comunicación o escuchándose para mantenerse sincronizados entre ellos.

Escenario Failover

Como se ha mencionado con anterioridad el escenario de *failover*, ocurre cuando se vuelve necesario cambiar al proveedor de datos o nodo maestro por otro nodo, a consecuencia de una falla en el nodo maestro.

Entonces es necesario que el nuevo nodo maestro se sincronice con los demás nodos que forman parte de la replicación, cambiando al nuevo origen del set de replicación.

El proceso de *failover* ocurre como se muestra en la figura 3.23:

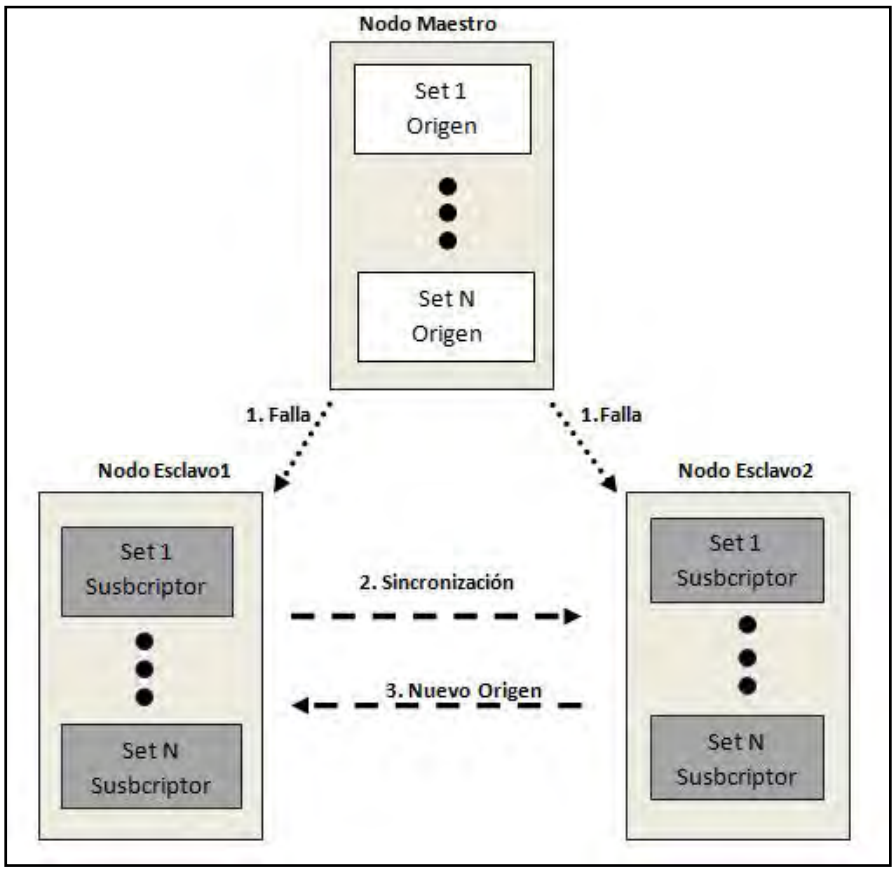


Figura 3.23 Escenario failover en el cluster.

A continuación se explican los pasos del proceso de *failover*:

1. El nodo maestro falla, como se muestra en la figura 3.23, este es el origen de datos del *set* 1 hasta el *set* N. El plan consiste en promover el nodo esclavo 1 como nodo maestro y dejar que el nodo esclavo 2 siga replicando con el nuevo nodo promovido como nodo maestro.

2. El nodo esclavo 1 primero preguntará por cada evento para ver si ya se encuentra replicado o no, mediante eventos SYNC para actualizar y sincronizar los nodos existentes.

3. Ahora al verificar la sincronización, el nodo esclavo 1 puede tomar su lugar y convertirse en el nuevo nodo origen.

Con la explicación del funcionamiento de los *sets* de replicación y el proceso de *failover,* se desea tener claros los principales procesos que comprenden la replicación dentro del *cluster*.

3.4.3 Diseño de sincronización

La sincronización en tiempo real de los nodos que conforman el *cluster*, es fundamental, porque de ello depende que los datos en los nodos sean los más actuales, por ello es necesario definir la forma en que se establecerá dicha sincronización.

En la sincronización del reloj de cada nodo utilizará el protocolo NTP, la utilización del protocolo asegurará que el reloj de tiempo real sea el mismo en los servidores para evitar generar errores en la sincronización de las réplicas.

En la figura 3.24 se muestra el arreglo de los nodos del *cluster*, como servidor NTP primario o *stratum1* se encuentra el nodo maestro que sincroniza su reloj con un servidor público de NTP específico para Norteamérica y en particular el de México, entonces con el reloj del nodo maestro sincronizado, este nos servirá para poder sincronizar el reloj de los nodos esclavo 1 y 2 que pertenecerán a un servidor NTP secundario o *stratum 2*, al tener un *stratum* tan corto la sincronización entre los nodos será muy precisa.

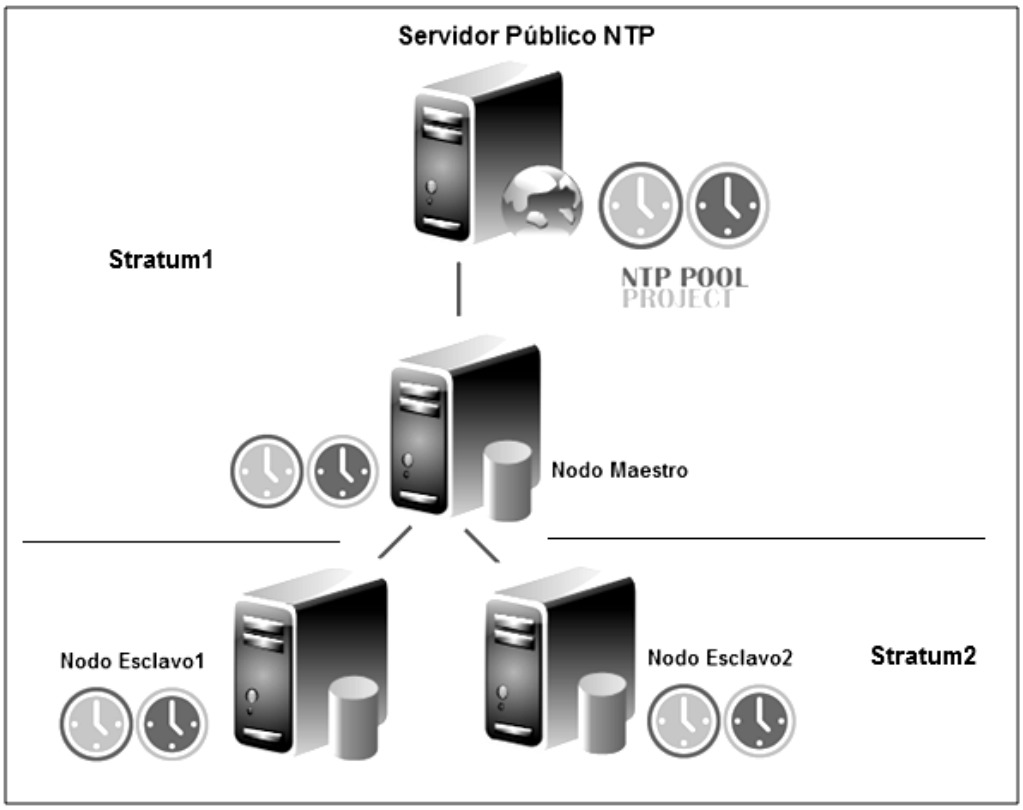


Figura 3.24 Diseño de sincronización de relojes.

3.4.4. Mecanismo de respaldo

En subcapítulos anteriores se mencionó la relevancia de realizar un esquema de respaldo por la importancia que representa la información dentro de una organización. La realización de respaldos permite tener de manera accesible los datos manejados por el sistema en el caso de que ocurriera una contingencia y fuera necesario restaurar las bases para poner de vuelta disponible el sistema.

Por ello, se realizan respaldos diarios de toda la base de datos dbgedgapa que contiene la información utilizada por los sistemas de la geDGAPA y el mecanismo que siguen dichos respaldos es el mostrado en la figura 3.25.

Figura 3.25 Mecanismo de respaldo.

El mecanismo de respaldo sigue la siguiente los siguientes pasos:

1. Desde un equipo se ejecuta una tarea automática de respaldo de la base de datos en el nodo maestro.
2. Realizado el respaldo, se extrae del nodo maestro y vuelve al equipo que ejecutó la tarea.
3. Ahora se procede a realizar la quema del respaldo.
4. Posteriormente los *dvds* que contienen los respaldos son almacenados.
5. Al tener los respaldos de un mes, es necesario llevarlos a una ubicación diferente de su origen, dicha acción es realizada por el responsable del traslado que los ubicará en el archivo general donde la información se encontrará salvaguardada.

De esa forma es como se manejan los respaldos de la geDGAPA, cabe mencionar que esto es un punto importante para implementar el *cluster,* porque se realiza una migración de datos de una versión a otra de PostgreSQL, donde se pone en juego la integridad de los datos y donde es conveniente tener un esquema de respaldo que respalde dicho proceso, si ocurriera un imprevisto y tuviera que restaurarse la base de datos anterior a la migración.

3.4.5 Diseño del proceso de migración

Como se ha explicado anteriormente, la etapa de migración consiste en ubicar los datos de las más de 800 tablas que se manejan en la geDGAPA a una versión más nueva de la utilizada actualmente de PostgreSQL, es decir, de la versión de PostgreSQL 7.4.6 a la versión de PostgreSQL 8.2.4, con el único fin de utilizar las ventajas que posee la nueva versión y de mantener el sistema actualizado.

En la figura 3.26 se observa de manera representativa, el proceso de migración de un servidor a otro, así como en la versión del DBMS.

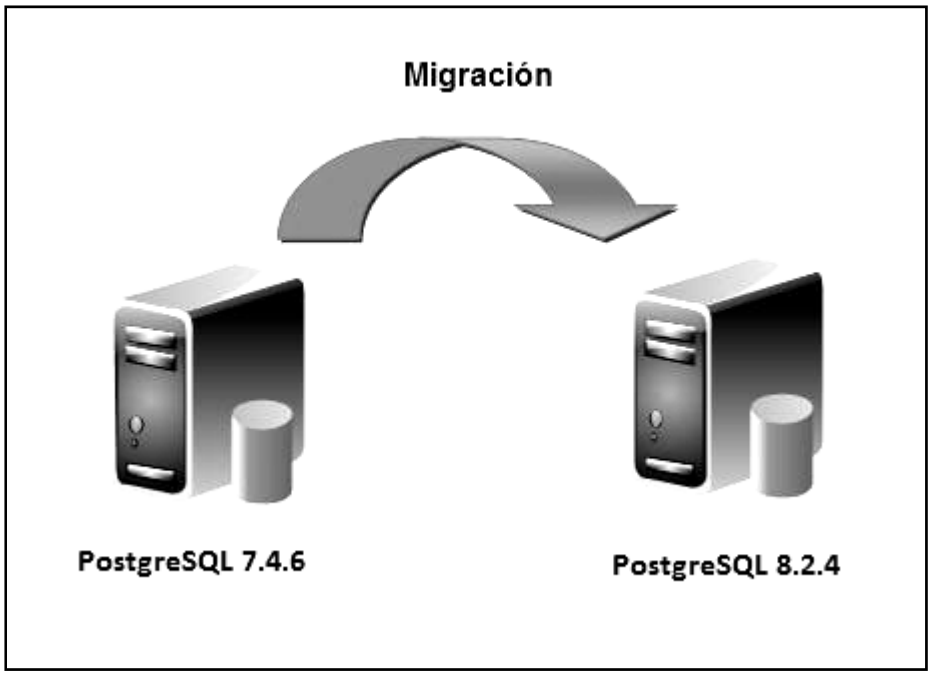


Figura 3.26 Diseño representativo del proceso de migración.

También es conveniente mencionar que dentro del proceso de migración se encuentran dos herramientas de gran utilidad, ya mencionadas con anterioridad, Data Comparer y DB Comparer.

Con el uso de las herramientas se logra tener al alcance una forma de comparar si los datos contenidos en las bases no se han dañado y en caso de que hubiera diferencias corregirlas antes de hacer la migración definitiva de la base.

En la figura 3.27 se muestra el lugar que ocupan las herramientas dentro del proceso de migración.

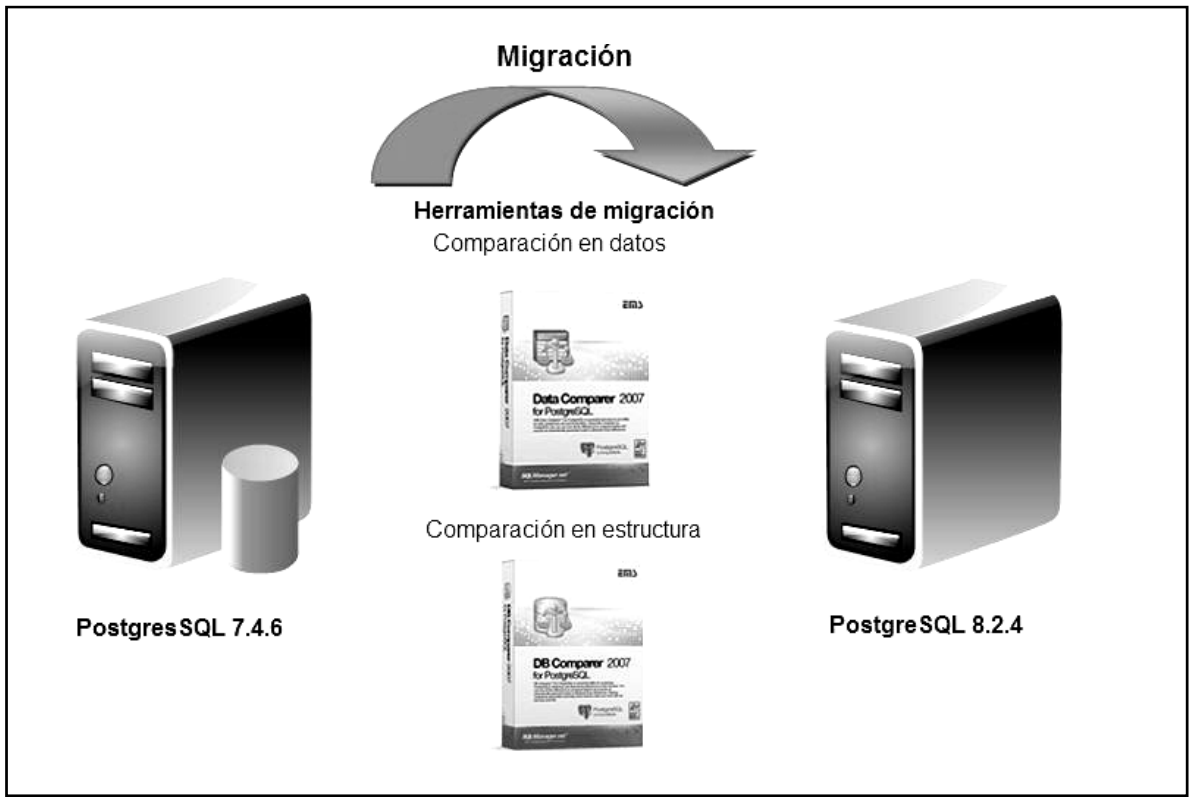


Figura 3.27 Herramientas utilizadas en el proceso de migración.

3.4.6 Diseño de las herramientas de monitoreo

El diseño de monitoreo para los componentes críticos del *cluster,* se tienen dos monitores que se observan en la figura 3.28 y que se enfocaran a los tres nodos del *cluster*.

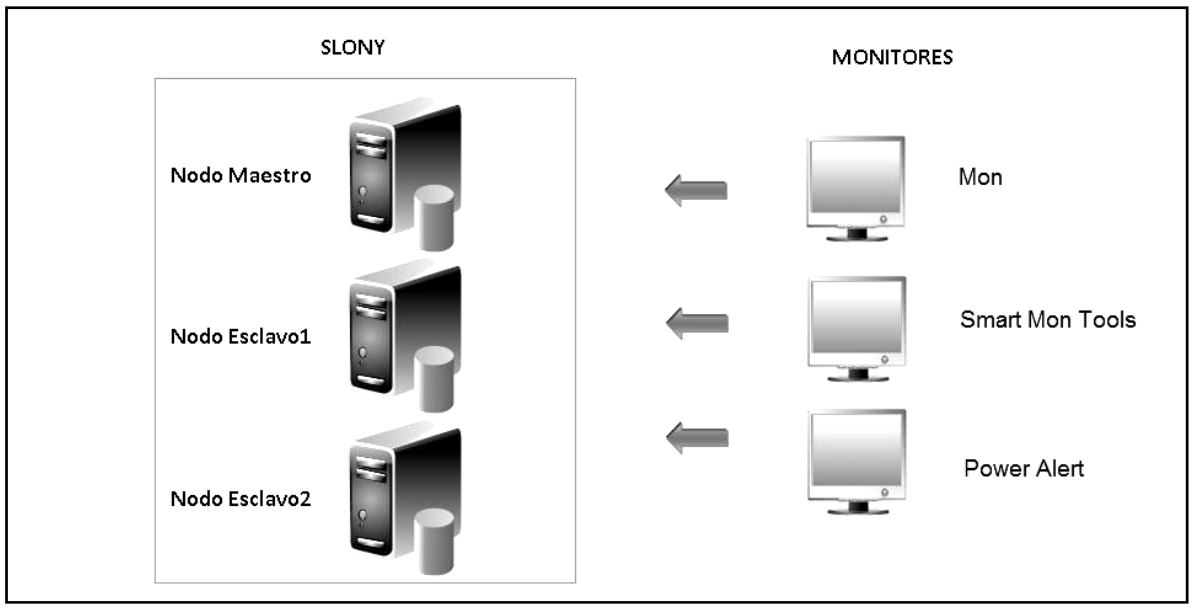


Figura 3.28 Monitores del cluster.

Mon tendrá el objetivo de cubrir que los tres nodos se encuentren arriba, es decir, en servicio, ya que Mon monitorea periódicamente según su configuración y en el caso de que alguno dejará de funcionar, mandará una alerta al administrador vía e-mail para que se tomen las medidas necesarias ante la problemática ocurrida.

Smart Mon Tools se encargará de verificar el estado de los discos duros de los servidores, periódicamente según se indique en su configuración, tiene la ventaja de detectar por anticipado posibles fallos físicos en los discos, ya que proporciona datos como la temperatura del disco, analizar el disco y descubrir si hay sectores dañados, saber la velocidad de los platos del disco, entre otros, todo ello para poder cambiar el disco duro que presenta problemas y evitar alguna contingencia.

Power Alert por su parte se encargará de monitorear el estado de energía de los UPS, cuando ocurra un fallo en el suministro eléctrico y entren en acción los UPS y alguno ya no tenga energía suficiente para alimentar el servidor, el Power Alert se encargará de apagarlo con el tiempo suficiente de la manera correcta para evitar que alguna de las descargas del suministro llegue a dañar algún componente del equipo.

El uso de estas herramientas tiene como fin evitar problemas que afecten el desempeño del *cluster*, prevenir o predecir fallos y tomar las medidas necesarias para evitar poner en riesgo los datos críticos que se manejan y para lo cual fue diseñado el *cluster*.

3.4.7 Diseño de la implementación del cluster

Para poder llegar a la parte de la implementación en este proyecto se tuvo que cumplir con tres etapas principales, que de manera breve mencionaré en este capítulo. La primera etapa es el análisis, seguido por el diseño y por último la implementación.

El análisis del proyecto se ha manejado desde los capítulos anteriores, lo que ha permitido establecer el planteamiento del problema, cuyo objetivo principal es la búsqueda de una solución integral de alto rendimiento y alta disponibilidad. Durante esta etapa, se revisó la situación actual de la dependencia en el área de gestión electrónica (geDGAPA), cuáles son sus requerimientos y qué es lo que espera obtener con la solución de alta disponibilidad.

De esta etapa se definió, que la mejor tecnología para la solución del problema, es la del *cluster* por considerarse un sistema de alta disponibilidad y por contar con la característica de la redundancia en sus componentes, para la protección y disponibilidad de la información.

Para la integración del *cluster* también se estableció que el sistema operativo a instalar en el *cluster* es Linux en su versión de Ubuntu Server 8, como DBMS se estableció PostgreSQL 8.2 y como *software* de replicación se eligió a Slony-1 1.2.15.

La etapa de implementación contempla la puesta en marcha y será el resultado del análisis y diseño previo, los cuales definirán las actividades a realizar para levantar el *cluster*.

El control de las actividades en la etapa de implementación ayuda a establecer el tiempo que tomará cada una de ellas. Las actividades establecidas, la descripción de la actividad y el tiempo contemplado para las mismas se muestran en la tabla 3.4.

Tabla 3.4 Esquema de planeación de actividades para la implementación del cluster.

Actividad	Descripción	Duración
Análisis de requerimientos	• Realizar un análisis de requerimientos para la implementación del *cluster* y el impacto que representa el mismo.	5 días
Investigación previa	• Recopilación de libros y documentos acerca del *cluster* y su implementación.	30 días
Planificación del proyecto	• Personas asignadas al desarrollo del proyecto. • Definición del objetivo y alcance del proyecto.	5 día
Diseño de la arquitectura del *cluster*	• Diseño de cada uno de los componentes que forman parte de *cluster*.	21 días
Implementación del *cluster*	• Instalación y configuración de cada uno de los componentes del *cluster*.	60 días
Pruebas	• Pruebas de funcionalidad del *cluster*, realizando pruebas de replicación, failover y sincronización.	15 días
Monitoreo	• Monitoreo del funcionamiento del *cluster*.	14 días
	Tiempo total del proyecto	**150 días**

Nota: El plan de trabajo detallado se especifica en un diagrama de Gantt en el Apéndice A.2.

Como se puede observar en la tabla 3.4 se tiene contemplado cierto tiempo para concluir cada una de las actividades mencionadas, pero este tiempo estará sujeto a cierta tolerancia de atraso a efecto de causas indirectas al proyecto (días inhábiles), considerando lo anterior, el tiempo total estimado para la conclusión del proyecto es de 150 días.

Capítulo 4
Implementación y pruebas

Capítulo 4 Implementación y pruebas

4.1 Introducción

En este capítulo se describen detalladamente las etapas de implementación del *cluster*, con las configuraciones necesarias para lograr su correcto funcionamiento, esto implica desde la instalación del sistema operativo hasta el ***tunning*** del *cluster*, con la finalidad de dejar claro al lector los pasos y pruebas que se realizaron en el mismo.

4.2 Instalación y configuración del cluster

El siguiente subcapítulo nos guiará a través de los pasos necesarios para implementar cada uno de los componentes del *cluster*.

4.2.1 Instalación y configuración del sistema operativo

La primera etapa consiste en la instalación del sistema operativo en los 3 servidores Sun Fire X4200. A continuación se explica el proceso detalladamente.

Para instalar Ubuntu server necesitamos seguir los siguientes pasos:

1. Descargar Ubuntu server 8 y quemar el *software* en un cd, disponible en:

   http://www.ubuntu.com/

2. Colocar el cd de instalación y ***bootear*** desde él, ejecutarlo y responder a las siguientes preguntas dentro el proceso de instalación de Ubuntu server:

   Lenguaje: English
   Ubicación: México
   Hostname: nodo1
   Nombre del dominio: dgapa.unam.mx

3. El instalador pregunta sobre la partición del disco, se elige el método guiado donde se crean las particiones automáticas, la de sistema (/) y la de ***swap***.

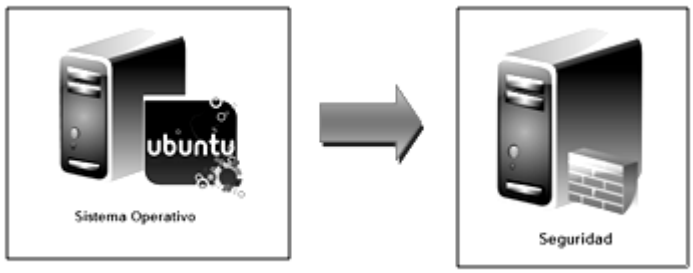
4. Se configura usuarios y contraseñas, entonces se le proporciona contraseña al administrador y al usuario común.

5. La instalación prosigue y finalmente el sistema ha sido instalado.

6. Finalmente después de reiniciar el equipo podremos entrar al sistema (figura 4.1):

Figura 4.1 Inicio del sistema operativo.

4.2.2 Instalación y configuración de la seguridad



Para que nuestro *cluster* tenga un desempeño óptimo es de vital importancia que cuente con seguridad propia, debido a las funciones de carácter crítico que desempeña.

La seguridad del *cluster* se controlará mediante *iptables*, en el capítulo anterior se proporcionó una introducción del concepto, uso y función de los *iptables*, con la finalidad de que en éste subcapítulo se entiendan los *scripts* de seguridad que se le aplicaron al *cluster*.

A continuación se muestra el *script* de seguridad del cluster, donde solo tienen acceso los equipos y puertos con los que se establece comunicación.

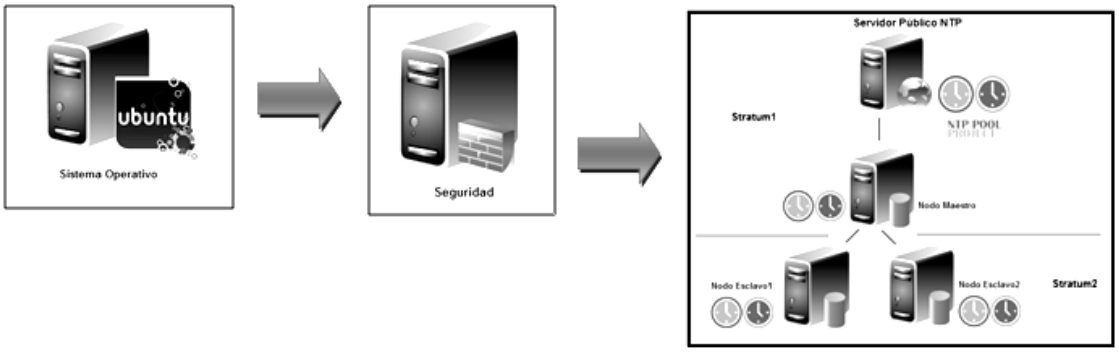
```
#!/bin/sh
iptables=/sbin/iptables
echo -n Aplicando Reglas de Firewall...

$iptables -t filter -F
$iptables -t nat -F

$iptables -t filter -P INPUT DROP
$iptables -t filter -P OUTPUT DROP
$iptables -t filter -P FORWARD DROP

$iptables -t filter -A INPUT -i lo -j ACCEPT
$iptables -t filter -A OUTPUT -o lo -j ACCEPT

#################### DNS DEL FIREWALL Y GATEWAY ################################

$iptables -t filter -A OUTPUT -o eth2 -p udp --dport 53 -d 132.248.204.1 -m state
--state NEW,ESTABLISHED -j ACCEPT

$iptables -t filter -A INPUT  -i eth2 -p udp --sport 53 -s 132.248.204.1 -m state
--state ESTABLISHED -j ACCEPT

$iptables -A FORWARD -o eth2 -p udp --dport 53 -d 132.248.204.1 -m state --state
NEW,ESTABLISHED -j ACCEPT
$iptables -A FORWARD -i eth2 -p udp --sport 53 -s 132.248.204.1 -m state --state
ESTABLISHED -j ACCEPT

###################### salida al firewall por puerto 80 #################

$iptables -t filter -A OUTPUT -o eth2 -p tcp --dport 80 -m state --state
NEW,ESTABLISHED -j ACCEPT
$iptables -t filter -A INPUT -i eth2 -p tcp --sport 80 -m state --state
ESTABLISHED -j ACCEPT

#########################################################################
# Acceso al firewall a las máquinas de los administradores
#########################################################################
#segmento  132.248.37.x

$iptables -A   INPUT -s 132.248.37.1   -p tcp --dport 22 -m state --state
NEW,ESTABLISHED -j ACCEPT
$iptables -A OUTPUT -d 132.248.37.1 -p tcp --sport 22 -m state --state
ESTABLISHED -j ACCEPT

$iptables -A   INPUT -s 132.248.37.10   -p tcp --dport 22 -m state --state
NEW,ESTABLISHED -j ACCEPT
$iptables -A OUTPUT -d 132.248.37.10 -p tcp --sport 22 -m state --state
ESTABLISHED -j ACCEPT

$iptables -A   INPUT -s 132.248.37.23   -p tcp --dport 22 -m state --state
NEW,ESTABLISHED -j ACCEPT
$iptables -A OUTPUT -d 132.248.37.23 -p tcp --sport 22 -m state --state
ESTABLISHED -j ACCEPT
```

```
$iptables -A   INPUT -s 132.248.37.171   -p tcp --dport 22 -m state --state
NEW,ESTABLISHED -j ACCEPT
$iptables -A OUTPUT -d 132.248.37.171 -p tcp --sport 22 -m state --state
ESTABLISHED -j ACCEPT

$iptables -A   INPUT -s 132.248.37.34   -p tcp --dport 22 -m state --state
NEW,ESTABLISHED -j ACCEPT
$iptables -A OUTPUT -d 132.248.37.34 -p tcp --sport 22 -m state --state
ESTABLISHED -j ACCEPT

#############################################################################
#Acceso a mon para http desde el servidor de desarrollo
#############################################################################
$iptables -A  INPUT -p tcp --dport 80 -m state --state NEW,ESTABLISHED -j ACCEPT
$iptables -A OUTPUT -p tcp --sport 80 -m state --state ESTABLISHED -j ACCEPT

########################## Monitor de Mon Reglas ICMP ##################
$iptables -A INPUT -i eth2 -p icmp -m state --state NEW,ESTABLISHED -j ACCEPT
$iptables -A OUTPUT -o eth2 -p icmp -m state --state ESTABLISHED -j ACCEPT

#############################################################################
#Acceso a la instancias de postgres    del nodo1 desde la maquina del
administrador2
#############################################################################
$iptables -A   INPUT -s 132.248.37.1   -p tcp --dport 5432 -m state --state
NEW,ESTABLISHED -j ACCEPT
$iptables -A OUTPUT -d 132.248.37.1 -p tcp --sport 5432 -m state --state
ESTABLISHED -j ACCEPT
#############################################################################
#Acceso al protocolo NTP con UDP con el puerto 123
#############################################################################
$iptables -A INPUT -i eth0 -p tcp --dport 123 -m state --state NEW,ESTABLISHED -j
ACCEPT
$iptables -A OUTPUT -o eth0 -p tcp --sport 123 -m state --state ESTABLISHED -j
ACCEPT
#############################################################################
#Acceso al nodo 2 externo del cluster
#############################################################################
$iptables -A OUTPUT -d 132.248.37.214   -p tcp --dport 5432 -m state --state
NEW,ESTABLISHED -j ACCEPT
$iptables -A INPUT -s 132.248.37.214   -p tcp --sport 5432 -m state --state
ESTABLISHED -j ACCEPT
#############################################################################
#Acceso al nodo 3 externo del cluster
#############################################################################
$iptables -A OUTPUT -d 132.248.37.38   -p tcp --dport 5432 -m state --state
NEW,ESTABLISHED -j ACCEPT
$iptables -A INPUT -s 132.248.37.38   -p tcp --sport 5432 -m state --state
ESTABLISHED -j ACCEPT
#############################################################################
```

Lo anterior es la configuración del nodo maestro del *cluster*, es necesario mencionar, que el nodo esclavo 1 y esclavo 2 cuentan con su propio *script* de seguridad, en general los tres nodos contienen los mismos permisos establecidos para comunicarse entre sí.

La configuración de los *iptables* actúan como *firewall*, en donde, solo poseen acceso los equipos y puertos necesarios, para evitar correr algún riesgo de seguridad que perjudique el desempeño del *cluster*.

4.2.3 Configuración de la sincronización del tiempo



Inmediatamente después de configurar la seguridad de los servidores, es necesario sincronizar el reloj de los mismos, en el momento de replicar se vuelve fundamental la exactitud del tiempo de los nodos porque al tener una diferencia en tiempo en un nodo esclavo, puede confundirse el demonio slon de la replicación pensando que va más adelantado en la réplica que el nodo original o maestro, lo que implicaría que el demonio slon tomará como un hecho que el nodo ya cuenta con los datos de esa diferencia de tiempo, por lo que terminará la replicación de manera incorrecta.

Para evitar este tipo de problemas se recurre al protocolo NTP, porque dicho protocolo permite mantener los nodos en constante comunicación para sincronizar el tiempo entre ellos y con ello asegurar una replicación correcta.

La instalación del protocolo NTP en un ambiente Linux se realiza de la siguiente manera:

1. Abrir una terminal.
2. Instalar el protocolo con:

   #apt-get install ntp

3. Ahora nos ubicamos en el archivo de configuración de NTP, que se encuentra en:

   #cd /etc/ntp.conf

   En el cual se configuran los servidores NTP públicos de donde se obtendrá la hora para sincronizar el nodo maestro del *cluster*.

```
# You do need to talk to an NTP server or two (or three).
server mx.pool.ntp.org
server 0.north-america.pool.ntp.org
server 1.north-america.pool.ntp.org
server 2.north-america.pool.ntp.org
server 3.north-america.pool.ntp.org
```

4. Para la configuración de los nodos esclavos, se debe especificar en su archivo ntp.conf que estos serán llamados *peers* del servidor local, es decir, del nodo maestro con el que se sincronizarán. Por lo que se agrega la siguiente línea:

```
# You do need to talk to an NTP server or two (or three).
peer 132.248.37.185
```

En la línea anterior se indica la ip de donde deben tomar la hora para sincronizarse.

5. Ahora se inicia el demonio de NTP desde la siguiente ubicación:

   #cd /etc/init.d

   Con el siguiente comando para iniciar el demonio de NTP:

   #./ntp start

   Cuando se requiera apagarlo en la misma ubicación se ejecuta:

   #./ntp stop

6. Para verificar que se encuentra funcionando el protocolo NTP ejecutamos los siguientes comandos en el nodo maestro :

   #ntpq -pn

   La salida que se obtiene es la siguiente:

```
     remote       refid         st t when poll reach   delay   offset  jitter
==============================================================================
*201.155.229.129 .GPS.            1 u  721 1024  177   34.459   -6.708  17.845
x208.53.158.34   198.186.191.229  3 u  708 1024  377  293.297  -83.533   2.115
+217.160.254.116 192.5.41.40      2 u  710 1024  377   99.408   13.850   0.622
+204.9.136.253   206.246.118.250  2 u  705 1024  377   86.558    7.128   1.072
-169.229.70.201  169.229.128.214  3 u  671 1024  177  303.695  117.486   4.195
```

   Donde se aprecia en la primera columna las direcciones ip´s de los servidores públicos de donde se puede obtener la hora para sincronizar, del lado izquierdo de la ip se observa un asterisco (*), el cual indica que es de ese servidor del que se está tomando la hora actualmente y con el que se sincroniza el nodo maestro, además se observa en la tercera columna el nivel de *stratum* (st) que presenta cada servidor y en la sexta columna (poll) se indica cada cuantos segundos se realiza la consulta con el servidor para mantenerse sincronizado.

   Si se desea conocer a detalle la información del servidor que se utiliza para sincronizar el reloj, se ejecuta el siguiente comando:

   #ntptrace

   Con el siguiente resultado:

```
localhost: stratum 2, offset 0.000402, synch distance 0.072032
dsl-201-155-229-129-sta.prod-empresarial.com.mx: stratum 1, offset
0.000001, synch distance 0.000451, refid 'GPS'
```

   Donde se observa las características del servidor local y del servidor NTP público, como el nivel de *stratum* de uno con respecto al otro, el tiempo de diferencia (*offset*) y los segundos que toma la distancia a la que se encuentran para la sincronización (*shych distance).*

En cambio para un nodo esclavo el resultado de ejecutar:
#ntpq -pn

Es el siguiente:

```
remote           refid            st t when poll reach   delay   offset  jitter
==============================================================================
*132.248.37.185  201.155.229.129  2 u    1  256  377    0.143   -0.380   1.826
```

Donde se observa que marca con asterisco (*) al nodo maestro, lo cual verifica que de él toma la hora para sincronizarse.

Y para el comando:

#ntptrace

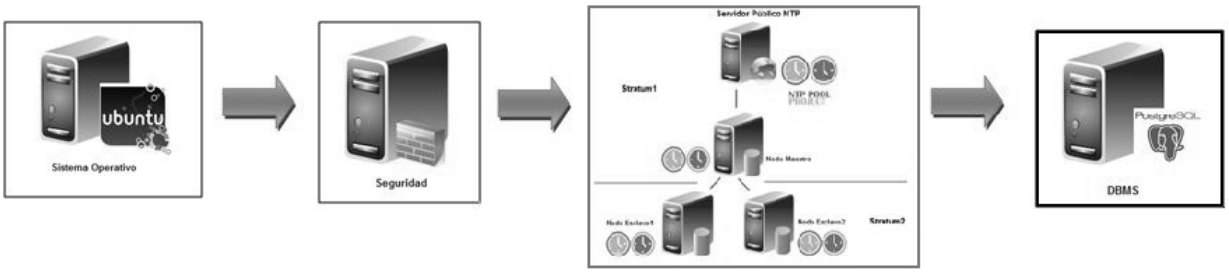
Se obtiene:

```
localhost: stratum 3, offset -0.002045, synch distance 0.086898
132.248.37.185: stratum 2, offset 0.000402, synch distance 0.074463
dsl-201-155-229-129-sta.prod-empresarial.com.mx: stratum 1, offset
-0.000003, synch distance 0.000312, refid 'GPS'
```

Donde se observa el nodo esclavo sincronizándose con el nodo maestro y este a su vez con el servidor NTP público, por el nivel de *stratum* que presenta cada uno.

Al utilizar el protocolo NTP se logra la sincronización entre los nodos con el fin de que la replicación se efectúe correctamente.

4.2.4 Instalación y configuración de PostgreSQL



Para instalar PostgreSQL en los nodos del *cluster*, estos deben de cumplir ciertos requerimientos mínimos para que el DBMS funcione correctamente, los cuales son:

- 8 *megabytes* de RAM.
- 30 *megabytes* de espacio en disco para el código fuente.
- 5 *megabytes* de espacio en disco para la instalación de ejecutables.
- 1 *megabyte* extra para las bases de datos básicas.[73]

[73] http://www.webexperto.com/articulos/articulo.php?cod=136
Instalación de un servidor de PostgreSQL bajo Linux por Luis Tomás Wayar.

Al cumplir los requerimientos anteriores, se prosigue con la instalación de PostgreSQL, a continuación se detallan los pasos a seguir:

1. Se descarga la versión 8.2.4 de la página oficial:

http://www.postgresql.org/

2. Una vez que tenemos el archivo fuente, se coloca en el directorio:
#cd /downloads/scripts_cluster/postgres

3. Ahora se crea la cuenta de ***superusuario*** de PostgreSQL, normalmente se usa por defecto como nombre de usuario "postgres" y se hace de la siguiente manera:

#adduser postgres
#passwd xxxxx

Ahora el usuario existe y toda la instalación del *software* y configuración se debe de hacer con la cuenta de “postgres”.

4. Es necesario que antes de empezar se lean los archivos README e INSTALL para ver los pasos de instalación que debemos seguir.

5. Para realizar la instalación de manera más rápida se crearon tres *scripts*:

1descomprime.sh
2install.sh
3afinacion_pg.sh

Ubicados en la misma ruta donde se colocó el archivo.tar de PostgreSQL.

6. Ahora como ***root*** ejecutamos el primer *script*:

#./1descomprime.sh

El cual contiene los comandos para descomprimir los archivos fuentes de PostgreSQL, como se muestra a continuación:

```
#!/bin/sh
#############################################################################
cd /downloads/scripts_cluster/postgres/install
tar zxvf postgresql-8.2.4.tar.gz && echo descomprimiendo codigos fuentes
de postgres
```

7. Ahora ejecutamos el segundo *script* donde se instala PostgreSQL:

#./2install.sh

```
#!/bin/sh
###############################################################################

echo instalar rapidamente Postgres
cd /downloads/scripts_cluster/postgres/install/postgresql-8.2.4
./configure && make && make install
echo Se ha instalado Postgres
adduser postgres && chown -R postgres.postgres /usr/local/pgsql
echo Se ha agregado el usuario de Postgres
&& mkdir /usr/local/pgsql/data
echo Se ha creado el directorio data
&& chown postgres.postgres /usr/local/pgsql/data
echo Se cambian los permisos del data para que su dueño sea postgres
```

8. Para finalizar la instalación se ejecuta el tercer *script*:

#./3afinacion_pg.sh

El objetivo de este *script* es continuar la configuración y afinar detalles que se requieren, como inicializar el dispositivo de datos, iniciar PostgreSQL y crear una base de datos de prueba.

```
#!/bin/sh
###############################################################################
#Para ejecutar el archivo se tiene que estar logeado como postgres
/usr/local/pgsql/bin/initdb -D /usr/local/pgsql/data &&
echo Inicializacion de un dispositivo de datos para postgres
pg_ctl start -l $PGDATA/server.log &&
echo arrancar el manejador de postgres con el dispositivo anterior y
todo el log al archivo logpostgres
/usr/local/pgsql/bin/createdb prueba &&
echo Se crea la BD de prueba llamada prueba y se entra a consola
/usr/local/pgsql/bin/psql prueba &&
\q && exit
echo Instalacion Finalizada
```

9. PostgreSQL necesita tener declaradas las siguientes variables de ambiente:

PGDATA=/usr/local/pgsql/data
PATH=/usr/local/pgsql/bin
MANPATH=/usr/local/pgsql/man

Dichas variables se agregan en el archivo .profile que se encuentra dentro del usuario postgres en:

#cd /home/postgres

#vim .profile

El archivo .profile con las variables de ambiente agregadas se muestra a continuación:

```
# ~/.profile: executed by the command interpreter for login shells.
PATH=/usr/local/pgsql/bin:$PATH:/usr/local/slony1-1.2.15/bin
export PATH
MANPATH=/usr/local/pgsql/man:$MANPATH
export MANPATH
PGDATA=/usr/local/pgsql/data
export PGDATA
PGMAIN=/usr/local/pgsql
export PGMAIN
export CLUSTERNAME=gedgapa
export MASTERDBNAME= dbgedgapa
export MASTERHOST=132.248.37.185
export REPLICATIONUSER=postgres
export PGBENCHUSER=pgbench
export SLAVEDBNAME1= dbgedgapa2
export SLAVEHOST1=132.248.37.214
export SLAVEDBNAME2= dbgedgapa3
export SLAVEHOST2=132.248.37.38
```

10. Ahora para que las variables sean tomadas en cuenta es necesario reiniciar el servicio de PostgreSQL, para ello se crea un *script* que tenga la tarea de reiniciar el servicio y se ejecuta con el siguiente comando:

#./start_pg.sh

```
#!/bin/sh
##############################################################################
#iniciar en el primer dispositivo demonio de postmaster
/usr/local/pgsql/bin/pg_ctl -D /usr/local/pgsql/data start -l
/usr/local/pgsql/data/server.log
```

Y de la misma manera es necesario tener un *script* donde se detenga el servicio de PostgreSQL:

#./stop_pg.sh

```
#!/bin/sh
##############################################################################
#detener en el primer dispositivo el demonio de postmaster
/usr/local/pgsql/bin/pg_ctl -D /usr/local/pgsql/data stop -l
/usr/local/pgsql/data/server.log
```

11. En la instalación de PostgreSQL se deben modificar los siguientes archivos propios del manejador:

pg_hba.conf - Archivo de autentificación para los usuarios permitidos.

postgresql.conf - Archivo de configuración de las opciones de PostgreSQL.

Ubicados en:

#cd /usr/local/pgsql/data

En el primero, pg_hba.conf, se configuran los equipos que tendrán acceso a PostgreSQL indicando a que bases de datos, con que usuario, mediante que ip y con qué método de autentificación, como se muestra a continuación:

```
# TYPE  DATABASE    USER        CIDR-ADDRESS          METHOD

# "local" is for Unix domain socket connections only
local   all         all                                     trust
# IPv4 local connections:
host    all         all         127.0.0.1/32            trust
# IPv6 local connections:
host    all         all         ::1/128                 trust
# IPv4 local connections:
host    all         all         132.248.x.x/32          trust
# IPv4 local connections:
#Acceso al nodo 1
host    all         all         132.248.x.x/32          trust
#Acceso al nodo 2
host    all         all         132.248.x.x/32          trust
#Acceso al nodo 3
host    all         all         132.248.x.x/32           trust
```

En el segundo, postgresql.conf, se configuran los parámetros específicos de PostgreSQL, a continuación se muestran parte del archivo de configuración donde se especifica que escucha a todas las direcciones y por el puerto que escucha:

```
#------------------------------------------------------------------------------
# CONNECTIONS AND AUTHENTICATION
#------------------------------------------------------------------------------

# - Connection Settings -

listen_addresses = '*'          # what IP address(es) to listen on;
                                        # comma-separated list of addresses;
                                        # defaults to 'localhost', '*' = all
                                        # (change requires restart)
port = 5432                             # (change requires restart)
max_connections = 100                   # (change requires restart)
# Note: increasing max_connections costs ~400 bytes of shared memory per
# connection slot, plus lock space (see max_locks_per_transaction).  You
# might also need to raise shared_buffers to support more connections.
#superuser_reserved_connections = 3     # (change requires restart)
#unix_socket_directory = ''             # (change requires restart)
```

12. Finalmente la instalación es concluida y se continúa con el siguiente componente a implementar en el *cluster*

Consola Web de PostgreSQL

Para administrar los servidores de bases de datos que se utilizan en el *cluster* se debe contar con una consola para el manejo de las mismas, de manera que resulte fácil acceder a ellas, para realizar operaciones de inserción, borrado, actualización, entre otras operaciones.

La consola a utilizar para administrar las bases de datos es phpPgAdmin-4.2.2 y es una herramienta *web* de interfaz gráfica para el usuario que permite administrar un servidor de bases de datos de PostgreSQL.

Lo anterior es de utilidad para administrar las bases de datos que serán replicadas en el *cluster*. A continuación se describen los pasos de instalación de phpPgAdmin:

1. Se descarga el archivo fuente de la página oficial de phpPgAdmin:

   http://phppgadmin.sourceforge.net/

2. Se coloca el archivo en:

   #cd /var/www/html

3. De ahí se procede a descomprimir el archivo con:

   #tar xzvf phpPgAdmin-4.2-Beta-1

4. Se modifica el *script* de configuración de phpPgAdmin que se encuentran ubicado en:

   #cd /var/www/html/conf

   Con el nombre de:

   config.inc.php

   En el cual se configuran cada uno de los nodos y se habilita el soporte para Slony para poder administrar los nodos desde la consola, a continuación se muestra el segmento del *script* con la configuración de los tres nodos del *cluster.*

```
//Nodo maestro
 $conf['servers'][1]['desc'] = 'nodo 1';
 $conf['servers'][1]['host'] = '132.248.37.185';
 $conf['servers'][1]['port'] = 5432;
 $conf['servers'][1]['sslmode'] = 'disable';
 $conf['servers'][1]['defaultdb'] = 'template1';
 $conf['servers'][1]['pg_dump_path'] = '/usr/local/pgsql/bin/pg_dump';
  $conf['servers'][1]['pg_dumpall_path'] =
'/usr/local/pgsql/bin/pg_dumpall';
 $conf['servers'][1]['slony_support'] = true;
 $conf['servers'][1]['slony_sql'] = '/usr/local/pgsql/share';

 //Nodo esclavo 1
$conf['servers'][2]['desc'] = 'nodo 2';
 $conf['servers'][2]['host'] = '132.248.37.214';
 $conf['servers'][2]['port'] = 5432;
 $conf['servers'][2]['sslmode'] = 'disable';
 $conf['servers'][2]['defaultdb'] = 'template1';
 $conf['servers'][2]['pg_dump_path'] = '/usr/local/pgsql/bin/pg_dump';
$conf['servers'][2]['pg_dumpall_path'] =
'/usr/local/pgsql/bin/pg_dumpall';
 $conf['servers'][2]['slony_support'] = true;
 $conf['servers'][2]['slony_sql'] = '/usr/local/pgsql/share';

//Nodo esclavo 2
$conf['servers'][3]['desc'] = 'nodo 3';
 $conf['servers'][3]['host'] = '132.248.37.38';
 $conf['servers'][3]['port'] = 5432;
 $conf['servers'][3]['sslmode'] = 'disable';
 $conf['servers'][3]['defaultdb'] = 'template1';
 $conf['servers'][3]['pg_dump_path'] = '/usr/local/pgsql/bin/pg_dump';
```

```
 $conf['servers'][3]['pg_dumpall_path'] =
'/usr/local/pgsql/bin/pg_dumpall';
 $conf['servers'][3]['slony_support'] = true;
 $conf['servers'][3]['slony_sql'] = '/usr/local/pgsql/share';
```

5. Con un explorador de internet se accede a phpPgAdmin, donde se visualizan los nodos del cluster y su contenido, como se observa en la figura 4.2.

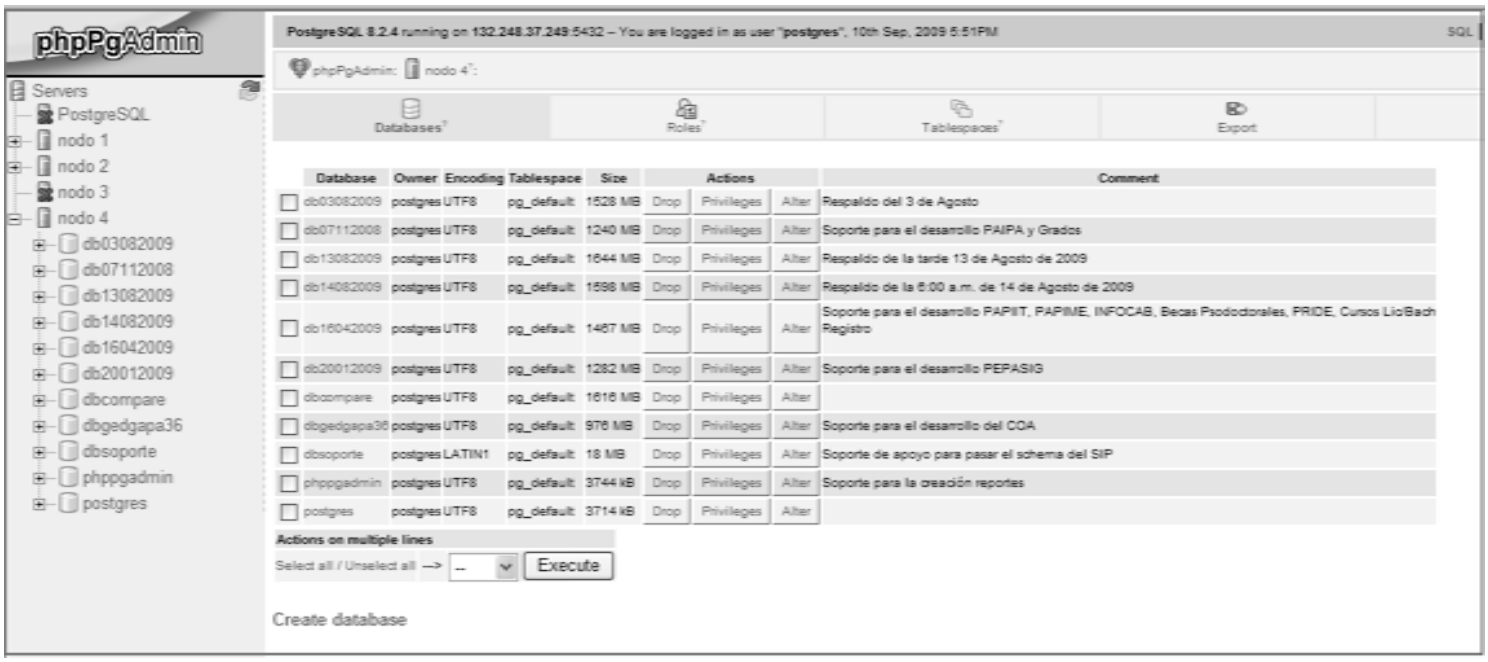


Figura 4.2 Vista de la consola web de phpPgAdmin.

Con lo anterior se puede controlar de forma fácil y practica las bases de datos utilizadas en la replicación dentro del *cluster*.

4.2.5 Proceso de respaldo

La realización de respaldos es parte fundamental para mantener de manera accesible la información cuando se requiera, por dicha razón es necesario implementar un proceso de respaldo que permita tener uso y control adecuado de los mismos.

Los sistemas que se manejan dentro de la geDGAPA accedan a una base principal dbgedgapa, por lo tanto, los respaldos que se hacen sobre dicha base tienen una ocurrencia de tres veces al día en los siguientes horarios:

1. A la 1:00 a.m.
2. A las 16:00 p.m.
3. A las 22:00 hrs

Para realizar los respaldos tres veces al día se ejecutan tareas automáticas con la herramienta de Apache ANT que permite realizar tareas mecánicas y repetitivas, a continuación se muestra un ejemplo de una tarea de respaldo:

```
<?xml version="1.0" encoding="UTF-8"?>
<project basedir="." default="init" name="respdia">
    <taskdef          name="ssh"          classname="com.sshtools.ant.Ssh"
classpath="C:\apache-ant-1.6.2\lib\maverick-ant.jar"/>

<target depends="init" name="3_1_resp_dia_185">
<echo>----- Inicia respaldo diario 132.248.37.185 -----</echo>
   <ssh       host="${host.linux.185}"       password="${passd.usr.pg.185}"
username="${usr.pg.185}">
            <exec     cmd="/usr/local/pgsql/bin/pg_dump     dbgedgapa     >
/home/postgres/3_dbgedgapa185_${now2}.dump"/>
            <exec    cmd="/usr/local/pgsql/bin/pg_dump    -s    dbgedgapa    >
/home/postgres/3_schemadbgedgapa185_${now2}.dump"/>
            <exec     cmd="/usr/local/pgsql/bin/pg_dump     phppgadmin     >
/home/postgres/3_phppgadmin185_${now2}.dump"/>
            <exec      cmd="/usr/local/pgsql/bin/pg_dump      dbayuda      >
/home/postgres/3_dbayuda185_${now2}.dump"/>
            <exec      cmd="/usr/local/pgsql/bin/pg_dump      dbantes      >
/home/postgres/3_dbantes185_${now2}.dump"/>
            <sftp action="get" remotedir="/home/postgres">
                <fileset dir="G:\${now}\">
                    <include name="*.dump"/>
                </fileset>
            </sftp>
            <exec cmd="rm *.dump"/>
        </ssh>
<echo>----- Fin respaldo diario 135.248.37.185 -----</echo>
</target>
```

El proceso interno de la tarea automática consiste en realizar la siguiente lista de pasos:

1. Verificar que exista suficiente espacio en la partición donde se generará el respaldo.

2. Para crear el respaldo, es necesario conectarse con la base de datos mediante el usuario postgres y ejecutar el siguiente comando:

#pg_dump dbgedgapa > /home/postgres/dbgedgapa_ddmmyyy_hhmm.dump

Donde se especifica la ubicación deseada para el respaldo y el formato del nombre del archivo, de tal manera que contenga el día, mes, año, hora y minuto de la creación del mismo, con el objetivo de crear una forma sencilla y rápida de localizar los respaldos para su clasificación.

3. Bajar el archivo "dbgedgapa_ddmmyyy_hhmm.dump" del servidor maestro al equipo encargado de realizar el respaldo.

4. Quemar el archivo en dvd para su resguardo y traslado mensual fuera de las instalaciones de la DGAPA.

Con esta lista de pasos se concluye el proceso de respaldo de la base dbgdgapa que interactúa con todos los sistemas manejados por la geDGAPA.

Así como es necesario un proceso de respaldo también es importante contar con un proceso de restauración del respaldo, ya que de nada servirá tener un respaldo si no se restaura adecuadamente.

Por lo tanto, proceso de restauración que se aplica en el caso de una contingencia, es el siguiente:

1. Crear una base de datos para la carga del respaldo:

CREATE DATABASE dbcarga ENCODING 'UNICODE' TEMPLATE template0

2. Sacar el respaldo de la base dbgedgapa, seleccionando el dia y el horario que se aproxime al tiempo de ocurrencia de la falla.

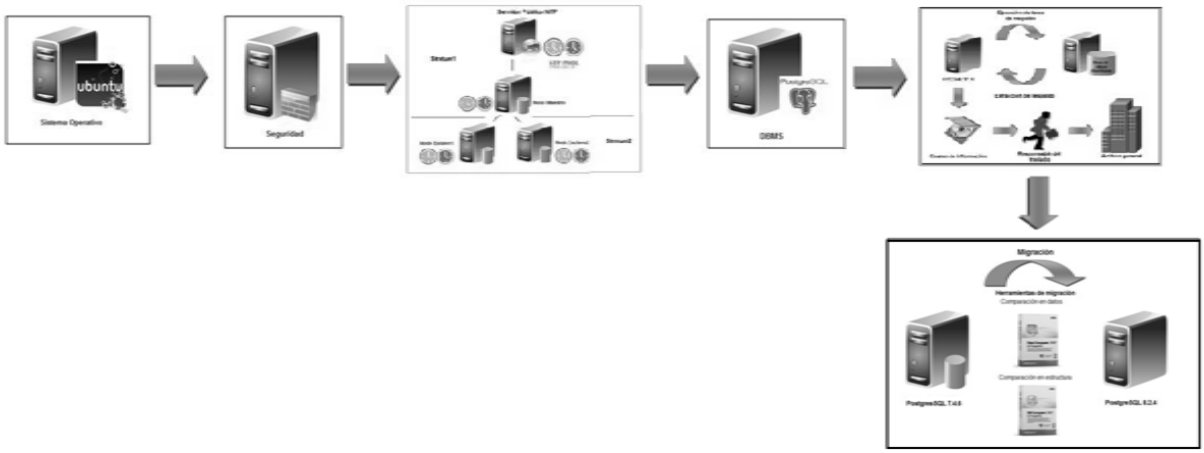
3. Ahora se carga el respaldo de la base seleccionado a un archivo que llamaremos “dbcarga” y se hace de la siguiente manera:

psql -e dbcarga < dbgedgapa_ddmmyyy_hhmm.dump

4. Finalmente se verifica la información contenida dbcarga para su restauración en la base principal.

Mediante el proceso de respaldo y restauración se cuenta con las estrategias necesarias para poner de nuevo en marcha los bases de datos que manejan las aplicaciones de la geDGAPA, este tipo de procesos aplican también a los respaldos que se realicen de las bases de datos de los nodos del *cluster*.

4.2.6 Migración



La migración es de gran importancia, debido al carácter crítico de los datos que contienen las bases de datos que participan en la migración, además es de las etapas que toman más tiempo definir y realizar, ya que se debe tomar en cuenta varios factores relevantes, como:

- El cambio en la sintaxis para definir estructuras de una versión a la otra.
- Realizar un listado de las bases de datos que son utilizadas por las aplicaciones de la geDGAPA, para no dejar ninguna exenta de migrarse.
- Realizar un respaldo del corte más actual de las bases de datos por cualquier problema que pudiera presentarse en el transcurso de la migración.
- Realizar un organigrama de las actividades a realizar desde apagar los servidores hasta volverlos a poner en línea.
- Realizar pruebas anteriores de migración antes de la definitiva, para prevenir errores y enriquecer el procedimiento final.

Los puntos anteriores son tan sólo los más primordiales antes de realizar la migración definitiva, además es necesario conocer los datos con los que se están trabajando, por ello, en la tabla 4.1 se observan los **esquemas** participantes en la migración con el número de tablas que contiene cada uno de ellos:

Tabla 4.1 Esquemas a migrar.

Esquemas	Descripción	N° de Tablas
actacad	Manejo de las tablas de actualización.	89
admin	Manejo de las tablas de la administración en geDGAPA.	17
aud	Manejo de las tablas de auditoría en geDGAPA.	5
audit_cat	Auditoría sobre catálogos.	1
audit_nucleo	Auditoria sobre el núcleo.	3

becas	Manejo de las tablas del asunto BECAS.	65
cat	Manejo de los catálogos del sistema geDGAPA.	84
coa	Manejo de las tablas del asunto COA.	18
dba	Manejo exclusivo del administrador de la base de datos. En este esquema se encuentran tablas y funciones para el mantenimiento de la base de datos dbgedgapa.	1
encuesta	Encuestas en geDGAPA.	8
grados	Manejo de las tablas del asunto GRADOS	13
nomina	Manejo de la nóminas del geDGAPA.	15
nucleo	Manejo de las tablas del núcleo del sistema geDGAPA.	41
paipa	Manejo de la información del asunto PAIPA.	35
papiit	Manejo de las tablas del asunto del PAPIIT en geDGAPA.	244
pepasig	Manejo de las tablas del asunto PEPASIG.	33
premios	Manejo de las tablas del asunto PREMIOS.	45
pride	Manejo de las tablas del asunto del PRIDE en geDGAPA.	64
public	Esquema *standard* público	118
segdoc	Manejo de las tablas del seguimiento de la documentación en geDGAPA.	8
sip	Sistema integral de personal.	22
syb	Relación con las base de datos en Sybase.	4
	Total	**933 tablas**

Para la migración de las bases de datos de la versión de PostgreSQL 7.4.6 a la 8.2.4, se enlistan los siguientes pasos:

1. Primero es necesario restringir el acceso a los usuarios de los sistemas de la geDGAPA, dejando entrar solo a los administradores del sistema y responsables de la migración.

2. Se realiza el respaldo de la base de datos del servidor a migrar de la siguiente manera:

   #pg_dumpall > dbgedgapa.dump

3. Ahora en el nuevo servidor se cargará el respaldo recién creado con:

   psql template1 < bases185.dump

4. Terminando el proceso de migración se puede proceder a comparar la estructura y datos de las bases de datos.

Instalación DB Comparer para PostgreSQL

La siguiente etapa es comparar la estructura de las bases de datos del servidor anterior contra la estructura del nuevo servidor, para verificar que las estructuras se hayan copiado exitosamente y en caso contrario, modificar la recién migrada para que sea idéntica a la anterior.

Por ello, es necesario instalar la herramienta DB Comparer, la cual se descarga en su versión de prueba desde la siguiente página:

http://sqlmanager.net/products/postgres/dbcomparer

Se descomprime y se instala con la ayuda del instalador de Windows.

Ya instalada la herramienta se configura para comparar la estructura siguiendo los pasos listados a continuación:

1. Se insertan los datos de ubicación del servidor que contiene los datos de origen migrados (figura 4.3).

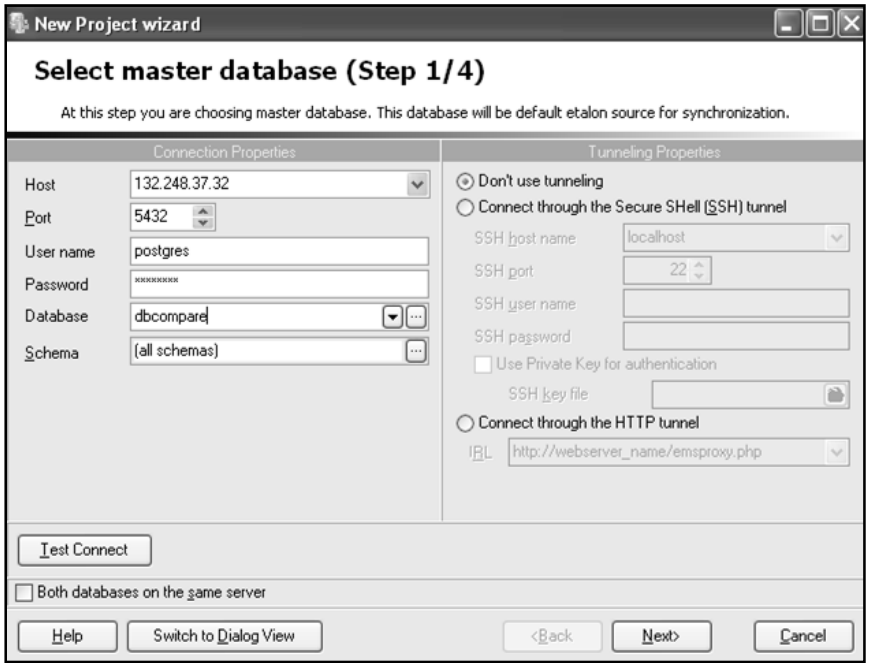


Figura 4.3 Configurando el servidor de bases de datos de origen.

2. Se insertan los datos de ubicación del nuevo servidor recién migrado (figura 4.4).

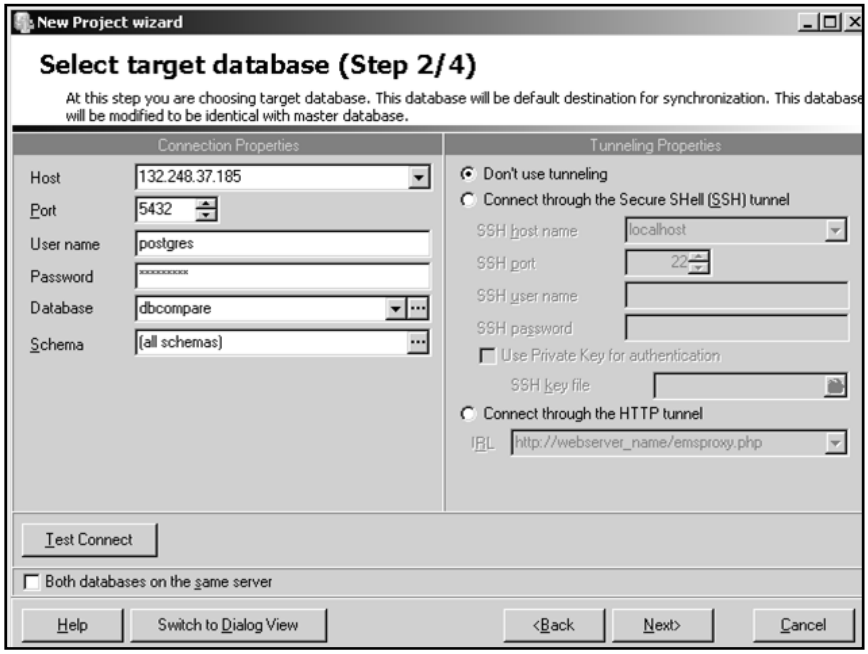


Figura 4.4 Configurando el servidor recién migrado.

3. Ahora se selecciona las características que se desee comparar (funciones, secuencias, tablas) entre ambos servidores de bases de datos (figura 4.5).

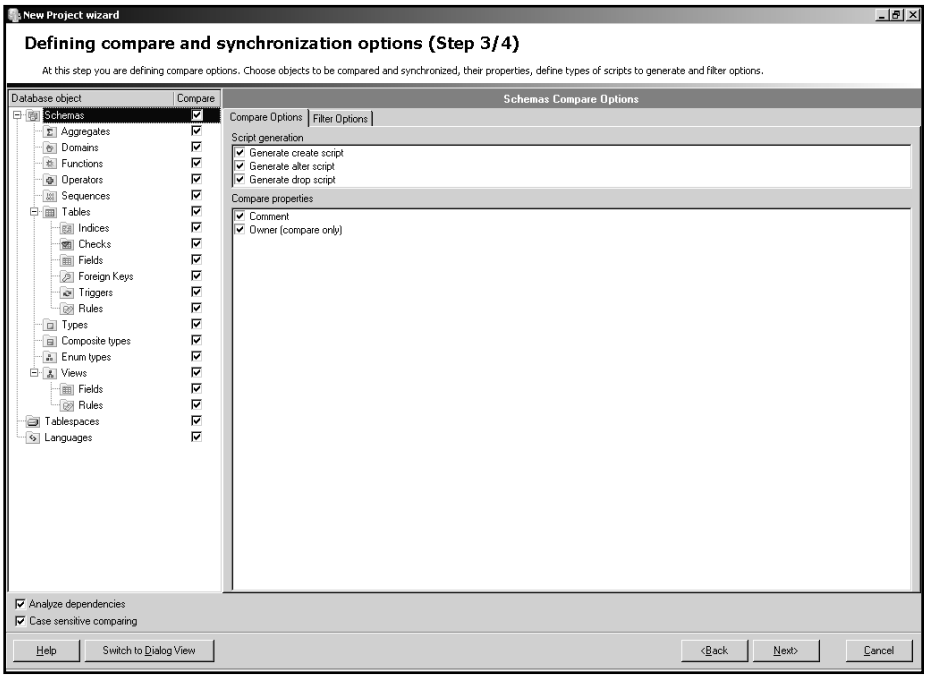


Figura 4.5 Seleccionando características a comparar.

4. Se activa el proceso de comparación (figura 4.6).

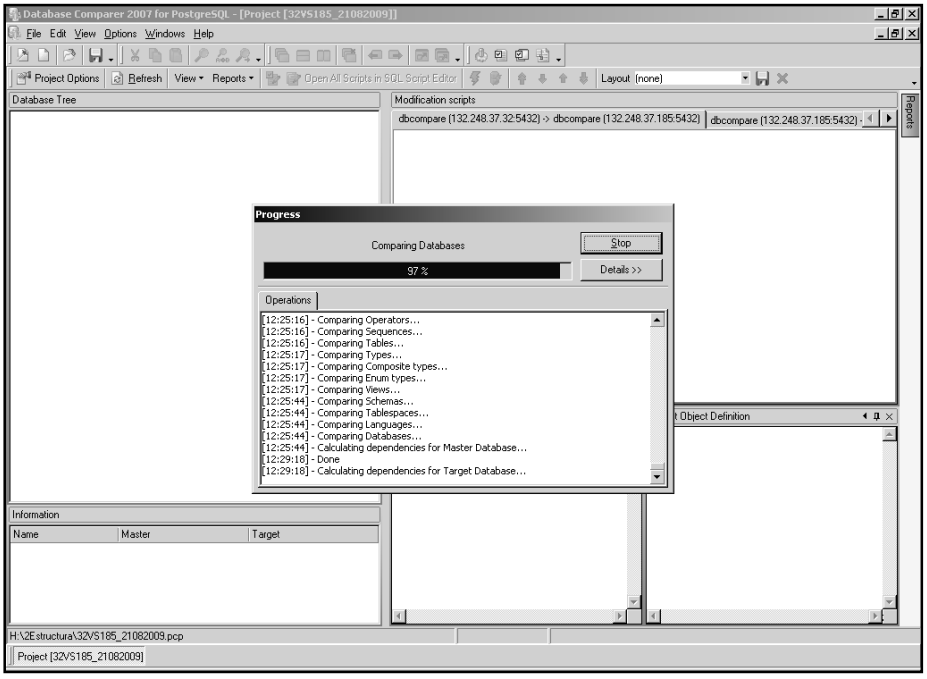


Figura 4.6 Ventana de ejecución de la comparación.

5. Al finalizar el proceso se muestra una ventana con toda la información de las diferencias y similitudes que se presentaron entre las bases de datos (figura 4.7).

Summary Info

Master database: dbcompare (132.248.37.249:5432)
Target database: dbcompare (132.248.37.32:5432)
Generated: 11/09/2009 12:07:15

Element	Different	Only Master	Only Target	Identical	Total
Aggregates	0	0	0	0	0
Domains	0	0	0	0	0
Functions	2	0	0	38	40
Operators	0	0	0	0	0
Sequences	0	0	0	720	720
Tables	642	0	0	291	933
Types	0	0	0	0	0
Composite types	0	0	0	0	0
Enum types	0	0	0	0	0
Views	33	0	0	397	430
Schemas	8	0	0	21	29
Tablespaces	0	0	0	0	0
Languages	0	0	0	1	1
Total:	685	0	0	1468	2153

Legend

Only Master: objects exist in dbcompare (132.248.37.249:5432) but there is no in dbcompare (132.248.37.32:5432)

Only Target: objects exist in dbcompare (132.248.37.32:5432) but there is no in dbcompare (132.248.37.249:5432)

Different: objects exist in both databases, but are different

Figura 4.7 Resultados del análisis con DB Comparer.

El resultado del análisis en estructura de las bases de datos se muestran 685 diferencias y fueron consecuencias de los cambios en la sintaxis de definir los *sequences* de las tablas de la versión de PostgreSQL 7.4.6 a la 8.2.4.

Posteriormente las diferencias fueron identificadas y arregladas para proseguir con la comparación de los datos contenidos entre ambas bases de datos de los servidores.

Instalación Data Comparer para PostgreSQL

Ahora se continúa con la instalación de Data Comparer, para comparar los datos contenidos en ambos servidores de bases de datos y observar si hay alguna diferencia entre ellos.

Para instalar Data Comparer se descarga la versión de prueba desde la siguiente página:

http://sqlmanager.net/products/postgres/datacomparer

Se descomprime y se instala con la ayuda del instalador de Windows.

Ya instalada la herramienta se comienza a utilizar y se configura para comparar los datos entre ambos servidores de bases de datos, siguiendo los pasos listados a continuación:

1. Se insertan los datos de ubicación de los servidores (figura 4.8):

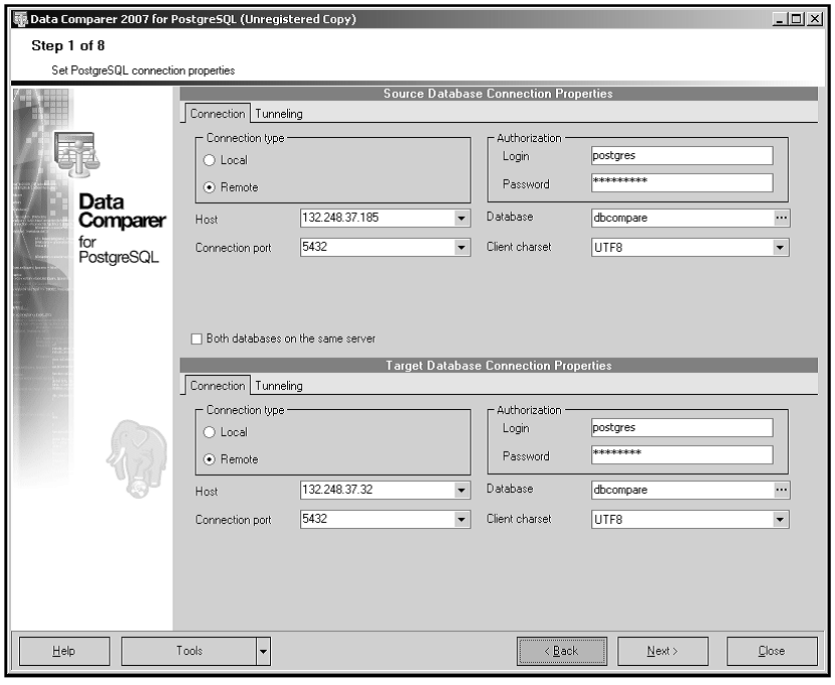


Figura 4.8 Captura de los datos de ambos servidores.

3. A continuación se seleccionan los esquemas que deseamos comparar (figura 4.9).

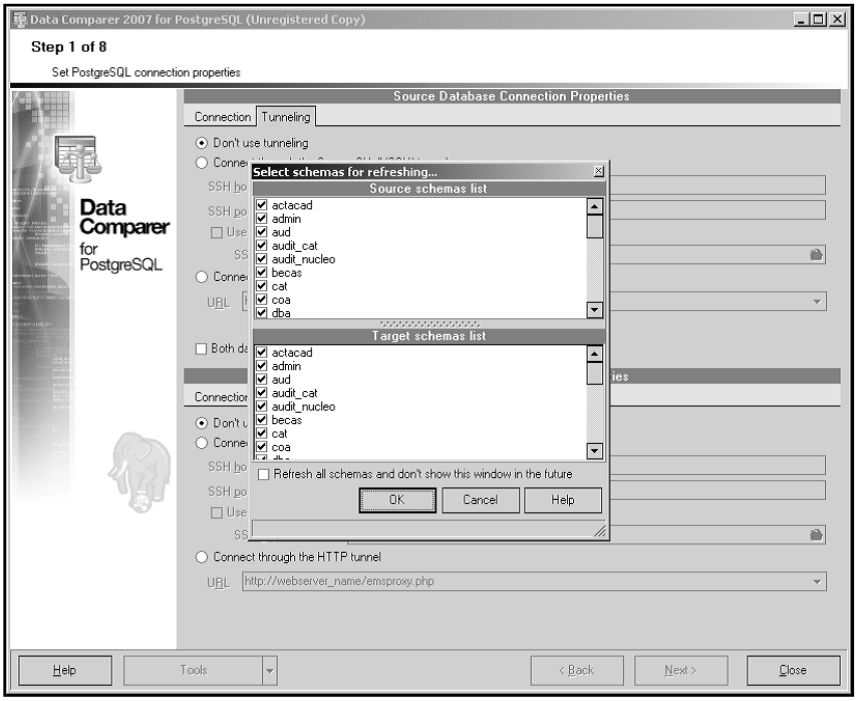


Figura 4.9 Selección de esquemas a comparar.

4. Ahora la herramienta compara los datos de los esquemas (figura 4.10).

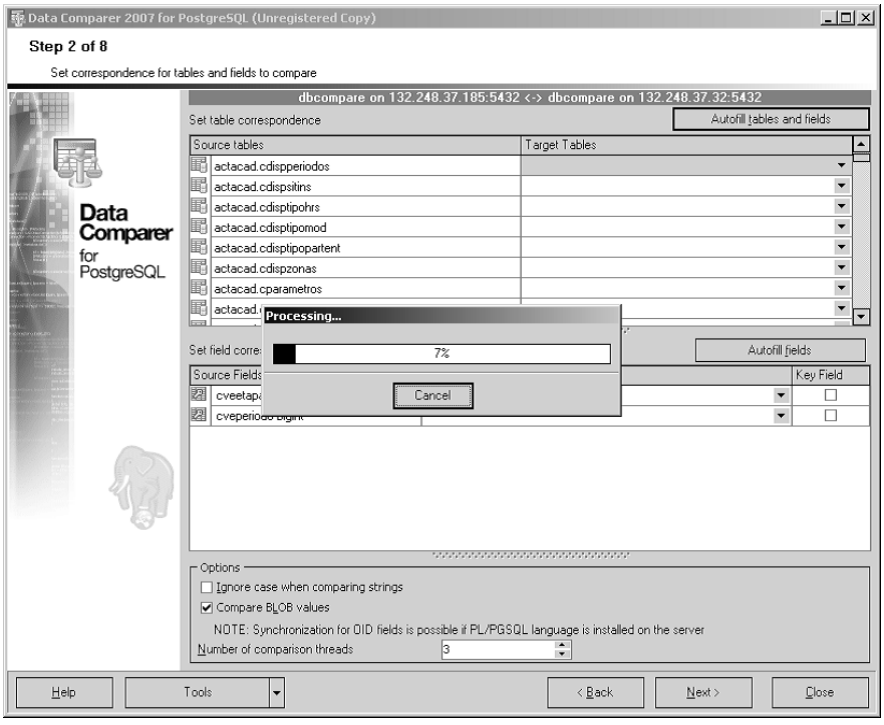


Figura 4.10 Comparando los datos de los esquemas seleccionados.

5. Concluido el análisis se generan tablas con los resultados, donde se muestran por columna el número de registros idénticos, diferentes, faltantes y adicionales. (figura 4.11):

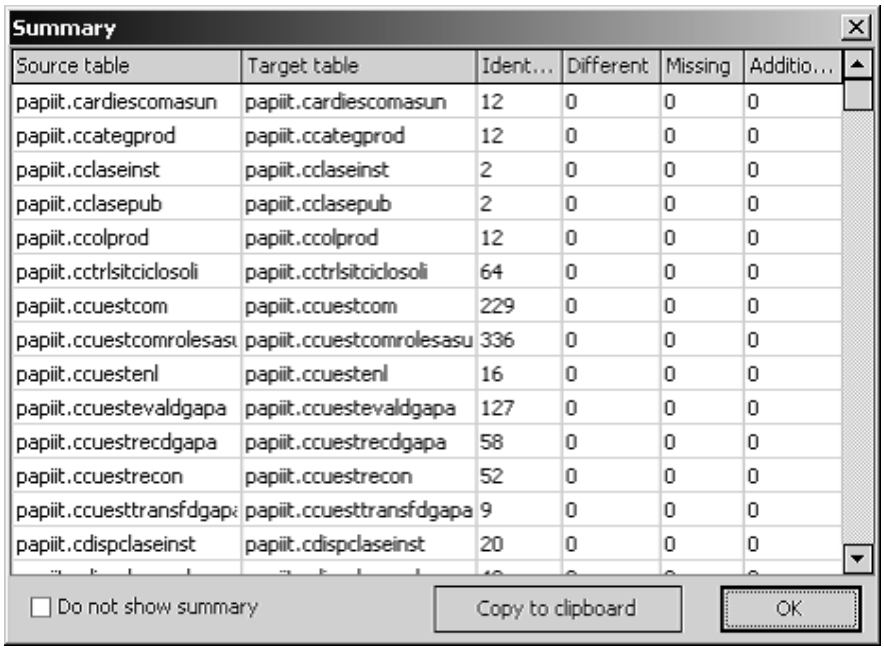
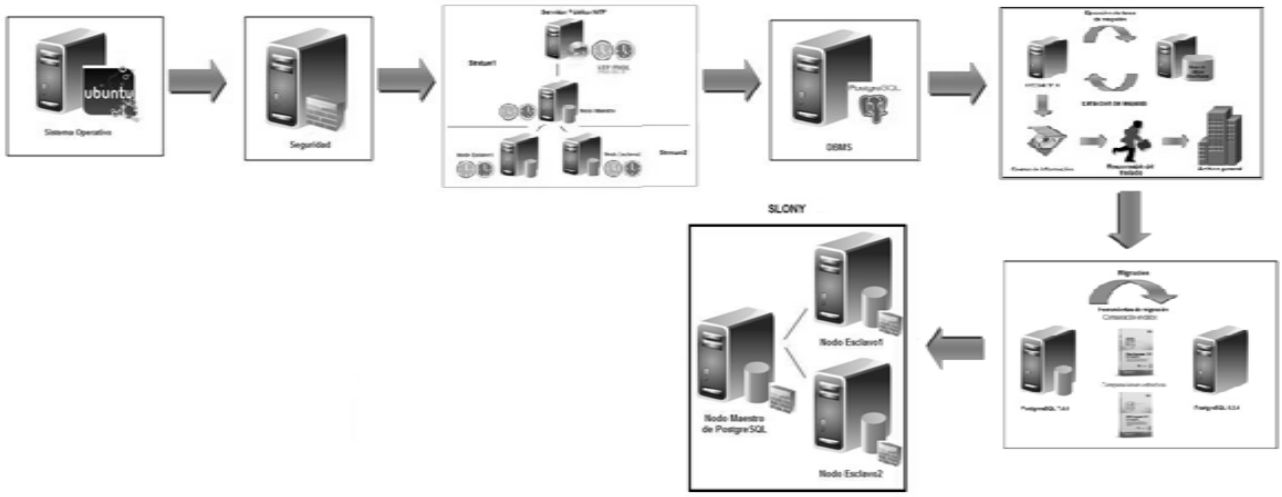


Figura 4.11 Resultados del análisis con Data Comparer.

Aquí observamos que no hay diferencias porque esta pantalla es una breve muestra de los resultados, con lo cual se concluyó que la migración fue un éxito.

4.2.7 Instalación y configuración de Slony-I



A continuación se describe el proceso de instalación del *software* Slony I-1.2.15, que será el replicador dentro del *cluster,* para fines prácticos lo llamaremos solamente Slony-I, este mismo proporciona la alta disponibilidad de las bases de datos utilizadas por la geDGAPA.

La instalación consiste en una serie de pasos, donde se especificará los elementos indispensables para que el *software* funcione adecuadamente y las configuraciones necesarias para nuestro sistema de replicación.

En principio, cualquier plataforma que pueda ejecutar PostgreSQL deberá ser capaz de ejecutar Slony-I.[74]

Para instalar Slony-I, es necesario tener previamente los siguientes componentes:

- GNU *make*. Es a menudo instalado bajo el nombre de *gmake*, es una herramienta que controla la generación de archivos ejecutables de un programa.
- Un compilador en lenguaje C.
- También se necesita una versión reciente de PostgreSQL, Slony-I depende del soporte de los ***namespace***, por lo tanto es necesario tener una versión de PostgreSQL 7.3.3 o posterior para poder construir y utilizar Slony-I.

Con los requisitos anteriores se procede a realizar los siguientes pasos para la instalación de Slony-I en el nodo maestro del *cluster*:

1. Bajar el archivo fuente de Slony-I desde la página oficial:

http://www.slony.info/

2. Ahora se coloca el archivo en:

#cd /downloads/scripts_cluster/slony

3. Descomprimir el archivo .tar con:

#tar xvf slony1-1.2.15.tar

4. Mover los archivos de Slony a la ruta:

#mv slony1-1.2.15 /usr/local/src

5. Ver los archivos de instalación README e INSTALL para conocer de manera general como realizar la instalación.

6. En la ruta donde previamente se movieron los archivos de Slony, ejecutamos el siguiente comando para empezar a cargar el *software*:

#./configure --prefix=/usr/local/src/slony1-1.2.15 --with-perltools

Nota:

- Si llegara a marcar el error de GNU *M4 1.4 missing* se realiza lo siguiente:

[74] http://www.slony.info/documentation/requirements.html
System Requirements of Slony-I

#apt-get install m4

- Si el error fuera *missing yacc parser.y parser.c* se realiza lo siguiente :

#apt-get install bison
#apt-get install flex

Que son herramientas complementarias para interpretar la sintaxis de Slony-I.

7. Prosiguiendo la instalación en la ruta usr/local/src se ejecuta el siguiente comando:

#make

8. De igual manera:

#make install

9. Nos regresamos un directorio atrás:

#cd /usr/local

10. Se crea una liga suave para redireccionar Slony1-1.2.15 a un nuevo directorio llamado slony1.

#ln -s slony1-1.2.15/ slony1

11. Ahora es necesario verificar en los archivos de configuración de PostgreSQL, pg_hba.conf y postgres.conf para poder establecer la comunicación entre PostgreSQL y los nodos de Slony-I.

12. Sí se realiza un cambio es necesario reiniciar los servicios de PostgreSQL para que se tomen los cambios realizados en cuenta.

13. Ahora se procederá a configurar los nodos de Slony-I con el usuario de postgres:

#su - postgres

14. Previamente se tiene un archivo respaldo generado de la migración de las bases de datos, entonces ahora se procede a cargar dicho respaldo para verlo en la consola de PostgreSQL.

#cd /downloads/scripts_cluster/dump
#psql dbgedgapa > archivo.dump

15. Ahora se instala el lenguaje pl/pgSQL en PostgreSQL de la siguiente manera con el usuario postgres:

#createlang -h localhost plpgsql dbgedgapa

16. Después de cargar el respaldo en el nodo maestro se hace una copia o volcado de la misma a los dos nodos esclavos, mediante el comando *dump*, para copiar la estructura y los datos de la base original y poder iniciar la replicación:

Para el nodo esclavo 1, se ejecuta lo siguiente:

```
#pg_dump -s -U $REPLICATIONUSER -h $MASTERHOST $MASTERDBNAME | psql
-U $REPLICATIONUSER -h $SLAVEHOST1 $SLAVEDBNAME1 -p 5432
```

Para el nodo esclavo 2, se ejecuta lo siguiente:

```
#pg_dump -s -U $REPLICATIONUSER -h $MASTERHOST $MASTERDBNAME | psql
-U $REPLICATIONUSER -h $SLAVEHOST2 $SLAVEDBNAME2 -p 5432
```

17. Ahora nos ubicamos en:

#cd /downloads/scripts_cluster/slony/escenario2

y ejecutamos el *script*:

#./2.maestroesclavo_config.sh

Donde se muestra la configuración de los primeros 5 sets de replicación de los 933 totales.

```
#!/bin/sh
#############################################################################
#Variables de ambiente tomadas por slony
#############################################################################
export CLUSTERNAME=gedgapa
export MASTERDBNAME=dbgedgapa
export MASTERHOST=132.248.37.185
export REPLICATIONUSER=postgres
export PGBENCHUSER=pgbench
export SLAVEDBNAME1=dbgedgapa2
export SLAVEHOST1=132.248.37.214
export SLAVEDBNAME2=dbgedgapa3
export SLAVEHOST2=132.248.37.38
#############################################################################
slonik <<_EOF_
       #--
       # define the namespace the replication system uses in our example it is
       # slony_example
       #--
       cluster name = $CLUSTERNAME;
       #--
       # admin conninfo's are used by slonik to connect to the nodes one for each
       # node on each side of the cluster, the syntax is that of PQconnectdb in
       # the C-API
       # --
       node 1 admin conninfo = 'dbname=$MASTERDBNAME host=$MASTERHOST
user=$REPLICATIONUSER';
       node 2 admin conninfo = 'dbname=$SLAVEDBNAME1 host=$SLAVEHOST1
user=$REPLICATIONUSER';
```

```
      node 3 admin conninfo = 'dbname=$SLAVEDBNAME2 host=$SLAVEHOST2
user=$REPLICATIONUSER';


      #--
      # init the first node.  Its id MUST be 1.  This creates the schema
      # _$CLUSTERNAME containing all replication system specific database
      # objects.

      #--
      init cluster ( id=1, comment = 'Master Node');


      #--
      # Because the history table does not have a primary key or other unique
      # constraint that could be used to identify a row, we need to add one.
      # The following command adds a bigint column named
      # _Slony-I_$CLUSTERNAME_rowID to the table.  It will have a default value
      # of nextval('_$CLUSTERNAME.s1_rowid_seq'), and have UNIQUE and NOT NULL
      # constraints applied.  All existing rows will be initialized with a
      # number
      #--
      #table add key (node id = 1, fully qualified name = 'public.history');

      #--
      # Slony-I organizes tables into sets.  The smallest unit a node can
      # subscribe is a set.  The following commands create one set containing
      # all 4 pgbench tables.  The master or origin of the set is node 1.


create set (id=1, origin=1, comment='replicando: actacad.cbanhrsdep');
set add table (set id=1 , origin=1, id=1, fully qualified name =
'actacad.cbanhrsdep', comment='replicando: actacad.cbanhrsdep');
set add sequence  (set id=1 , origin=1, id=2, fully qualified name =
'actacad.cbanhrsdep_cvebanhrsdep_seq', comment='replicando sequence:
actacad.cbanhrsdep_cvebanhrsdep_seq ');

create set (id=2, origin=1, comment='replicando: actacad.ccalificaciones');
set add table (set id=2 , origin=1, id=3, fully qualified name =
'actacad.ccalificaciones', comment='replicando: actacad.ccalificaciones');
set add sequence  (set id=2 , origin=1, id=4, fully qualified name =
'actacad.ccalificaciones_cvecalif_seq', comment='replicando sequence:
actacad.ccalificaciones_cvecalif_seq ');

create set (id=3, origin=1, comment='replicando: actacad.ccuestevalcom');
set add table (set id=3 , origin=1, id=5, fully qualified name =
'actacad.ccuestevalcom', comment='replicando: actacad.ccuestevalcom');
set add sequence  (set id=3 , origin=1, id=6, fully qualified name =
'actacad.ccuestevalcom_cvecuestevalcom_seq', comment='replicando sequence:
actacad.ccuestevalcom_cvecuestevalcom_seq ');

create set (id=4, origin=1, comment='replicando: actacad.ccuestevaldgapa');
set add table (set id=4 , origin=1, id=7, fully qualified name =
'actacad.ccuestevaldgapa', comment='replicando: actacad.ccuestevaldgapa');
set add sequence  (set id=4 , origin=1, id=8, fully qualified name =
'actacad.ccuestevaldgapa_cvecuestevaldgapa_seq', comment='replicando sequence:
actacad.ccuestevaldgapa_cvecuestevaldgapa_seq ');

create set (id=5, origin=1, comment='replicando: actacad.ccuestins');
set add table (set id=5 , origin=1, id=9, fully qualified name =
'actacad.ccuestins', comment='replicando: actacad.ccuestins');
set add sequence  (set id=5 , origin=1, id=10, fully qualified name =
'actacad.ccuestins_cvecuestins_seq', comment='replicando sequence:
actacad.ccuestins_cvecuestins_seq ');
```

```
        #--
        # Create the second node (the slave) tell the 2 nodes how to connect to
        # each other and how they should listen for events.
        #--

        store node (id=2, comment='Nodo esclavo_1');
        store node (id=3, comment='Nodo esclavo_2');

        store path (server=1, client=2, conninfo='dbname=$MASTERDBNAME
host=$MASTERHOST user=$REPLICATIONUSER');
        store path (server=1, client=3, conninfo='dbname=$MASTERDBNAME
host=$MASTERHOST user=$REPLICATIONUSER');

        store path (server=2, client=1, conninfo='dbname=$SLAVEDBNAME1
host=$SLAVEHOST1 user=$REPLICATIONUSER');
        store path (server=2, client=3, conninfo='dbname=$SLAVEDBNAME1
host=$SLAVEHOST1 user=$REPLICATIONUSER');

        store path (server=3, client=1, conninfo='dbname=$SLAVEDBNAME2
host=$SLAVEHOST2 user=$REPLICATIONUSER');
        store path (server=3, client=2, conninfo='dbname=$SLAVEDBNAME2
host=$SLAVEHOST2 user=$REPLICATIONUSER');


_EOF_
```

18. Ahora se levantan los demonios de Slony-I, con el siguiente comando para cada nodo:

Nodo Maestro
#./3demonio_maestro.sh

El cual contiene el comando para iniciar el demonio slon en el nodo maestro y a su vez la ruta donde se crea un archivo log para observar los mensajes de salida de ese nodo.

```
#!/bin/sh
#############################################################################
#Variables de ambiente tomadas por slony
#############################################################################
export CLUSTERNAME=gedgapa
export MASTERDBNAME=dbgedgapa
export MASTERHOST=132.248.37.185
export REPLICATIONUSER=postgres
#############################################################################
/usr/local/slony1-1.2.15/bin/slon  $CLUSTERNAME "dbname=$MASTERDBNAME
user=$REPLICATIONUSER host=$MASTERHOST" >
/var/log/slony1/archivelogs/slonNodo1.log 2>&1 &
```

Nodo Esclavo1
#./4demonio_esclavo1.sh

De igual manera el *script* para el nodo esclavo 1 contiene el comando para levantar el demonio slon y la ruta para generar el archivo log para los mensajes de salida.

```
#!/bin/sh
#############################################################################
#Variables de ambiente tomadas por slony
```

```
############################################################################
export CLUSTERNAME=gedgapa
export SLAVEDBNAME1= dbgedgapa2
export SLAVEHOST1=132.248.37.214
export REPLICATIONUSER=postgres
############################################################################

/usr/local/slony1-1.2.15/bin/slon  $CLUSTERNAME "dbname=$SLAVEDBNAME1
user=$REPLICATIONUSER host=$SLAVEHOST1 port=5432" >
/var/log/slony1/archivelogs/slonNodo2.log 2>&1 &
```

Nodo Esclavo2

#./5demonio_esclavo2.sh

Para el nodo esclavo 2 similarmente tenemos el demonio slon y la ruta del log de salida.

```
#!/bin/sh
############################################################################
#Variables tomadas por slony
############################################################################
export CLUSTERNAME=gedgapa
export SLAVEDBNAME2= dbgedgapa3
export SLAVEHOST2=132.248.37.38
export REPLICATIONUSER=postgres
############################################################################
/usr/local/slony1-1.2.15/bin/slon  $CLUSTERNAME "dbname=$SLAVEDBNAME2
user=$REPLICATIONUSER host=$SLAVEHOST2" >
/var/log/slony1/archivelogs/slonNodo3.log 2>&1 &
```

19. Entonces se procede a levantar la replicación con el siguiente *script*:

#./6start_replication.sh

El cual contiene la comunicación entre los nodos por medio de los *subscribe set*, que tiene como parámetros el id del set a replicar, el proveedor que es el nodo maestro y el subscriptor según sea el número del nodo esclavo y por último el *forward=yes* que indica que los nodos esclavos pueden cambiar de rol a nodo maestro en el proceso de *failover*.

Aquí solo se muestran los primeros cinco *subscribe sets* del *script.*

```
#!/bin/sh
############################################################################
#Variables tomadas por slony
############################################################################
export CLUSTERNAME=gedgapa
export MASTERDBNAME=dbgedgapa
export MASTERHOST=132.248.37.185
export REPLICATIONUSER=postgres
export PGBENCHUSER=pgbench
export SLAVEDBNAME1=dbgedgapa2
export SLAVEHOST1=132.248.37.214
export SLAVEDBNAME2=dbgedgapa3
export SLAVEHOST2=132.248.37.38
############################################################################
```

```
/usr/local/slony1-1.2.15/bin/slonik <<_EOF_
       # ----
       # This defines which namespace the replication system uses
       # ----
       cluster name = $CLUSTERNAME;
       # ----
       # Admin conninfo's are used by the slonik program to connect
       # to the node databases.  So these are the PQconnectdb arguments
       # that connect from the administrators workstation (where
       # slonik is executed).
       # ----
      node 1 admin conninfo = 'dbname=$MASTERDBNAME host=$MASTERHOST
user=$REPLICATIONUSER';
      node 2 admin conninfo = 'dbname=$SLAVEDBNAME1 host=$SLAVEHOST1
user=$REPLICATIONUSER';
      node 3 admin conninfo = 'dbname=$SLAVEDBANME2 host=$SLAVEHOST2
user=$REPLICATIONUSER';

      # ----
      # Node 2 subscribes set 1
      # ----
      subscribe set ( id = 1, provider = 1, receiver = 2, forward = yes);
      subscribe set ( id = 2, provider = 1, receiver = 2, forward = yes);
      subscribe set ( id = 3, provider = 1, receiver = 2, forward = yes);
      subscribe set ( id = 4, provider = 1, receiver = 2, forward = yes);
      subscribe set ( id = 5, provider = 1, receiver = 2, forward = yes);

      #---
      #Nodo 3 subscribes set 1
      #---
      subscribe set ( id = 1, provider = 1, receiver = 3, forward = yes);
      subscribe set ( id = 2, provider = 1, receiver = 3, forward = yes);
      subscribe set ( id = 3, provider = 1, receiver = 3, forward = yes);
      subscribe set ( id = 4, provider = 1, receiver = 3, forward = yes);
      subscribe set ( id = 5, provider = 1, receiver = 3, forward = yes);
_EOF_
```

20. Finalmente se checa en la consola de phpPgAdmin el estado de los tres nodos, que debe ser saludable como se muestra en la figura 4.12:

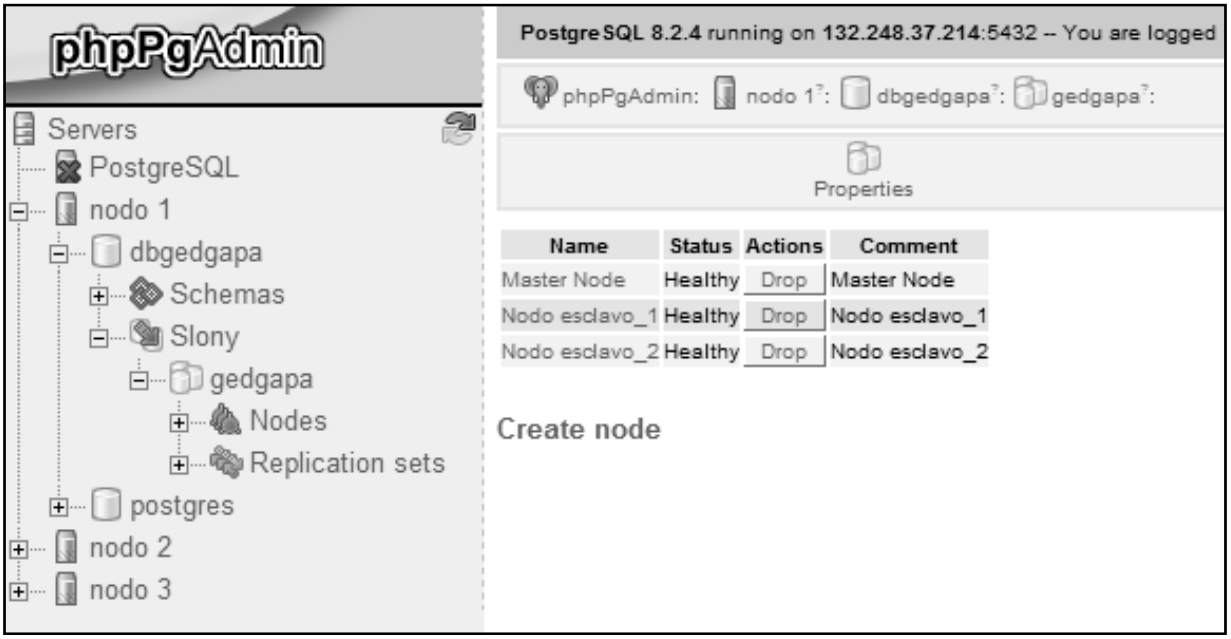


Figura 4.12 Vista del estado de los nodos del *cluster*.

También se puede observar en la consola de phpPgAdmin los *sets* que se configuraron para replicar con Slony-I (figura 4.13).

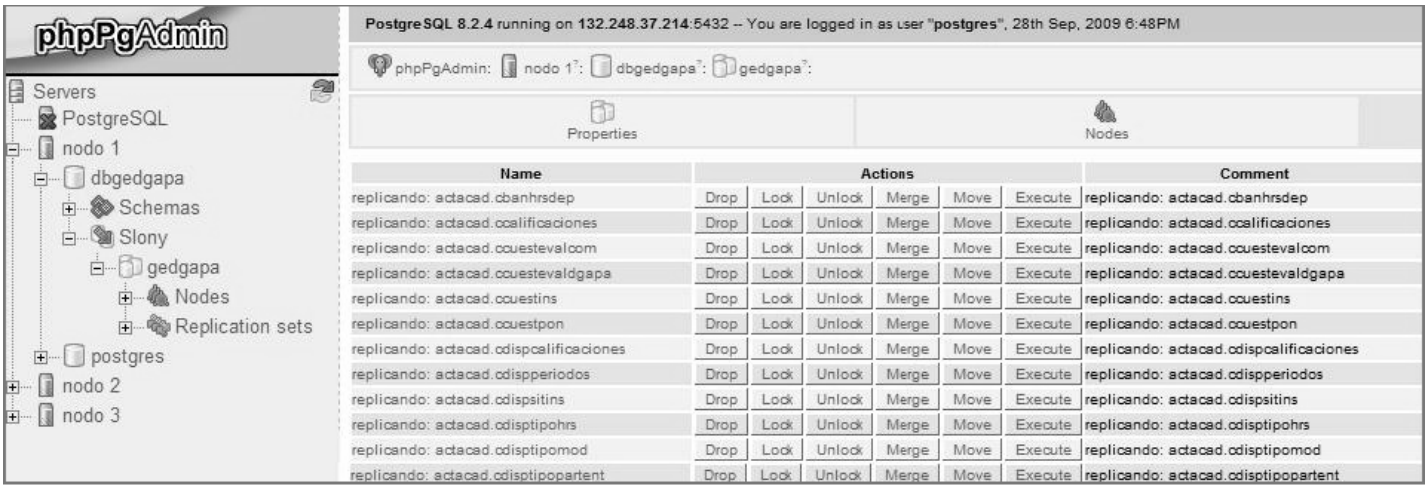


Figura 4.13 Vista de los sets de replicación del *cluster*.

21. Si se desea detener la replicación se ejecuta el siguiente *script*:

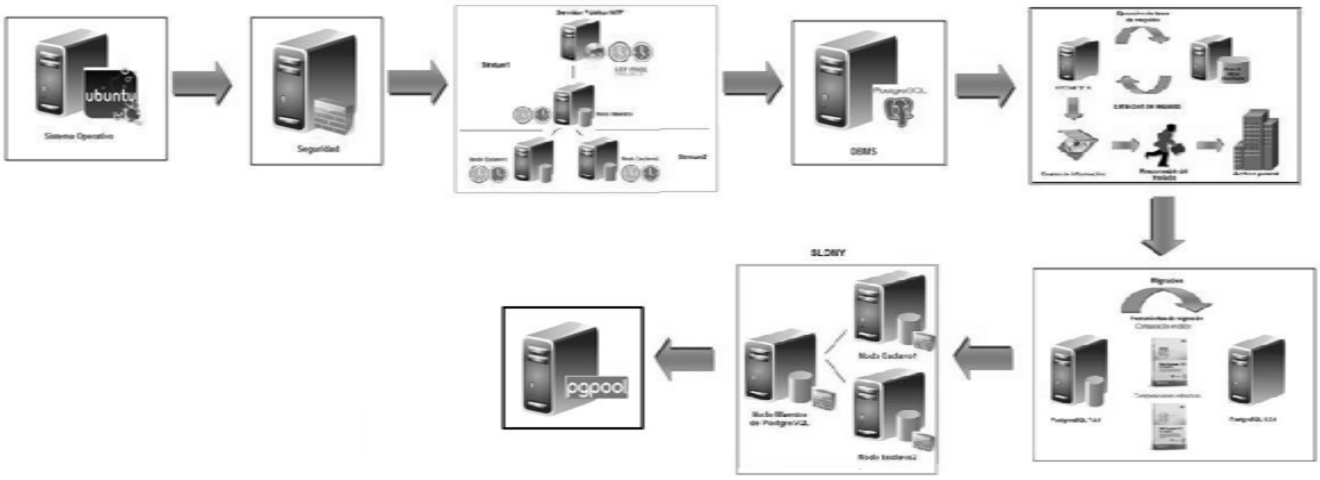
#./7slon_stop.sh

En el *script* se manda a llamar al demonio slon_kill para matar todos los procesos de Slony.

```
#!/bin/sh
################################################################################
/usr/local/slony1-1.2.15/bin/slon_kill
```

4.2.8 Instalación y configuración de Pgpool-II

Instalación Pgpool-II



Después de haber instalado Slony-I, el siguiente paso es instalar el *software* Pgpool-II que realizará el proceso de *failover* dentro del *cluster,* este proceso ocurre cuando el nodo maestro presenta alguna falla en su funcionamiento sea de *software* o de *hardware* que lo ponga fuera de servicio, por lo tanto , es necesario que uno de los dos nodos esclavos ocupe su lugar dentro de la replicación, es ahí donde Pgpool-II automáticamente hará el cambio por nosotros, entonces el sistema seleccionará el nodo a actuar como maestro para poder seguir replicando con la finalidad de tener siempre disponibles las bases de datos que manejan los sistemas dentro de la geDGAPA.

A continuación se listarán los pasos necesarios para la instalación de Pgpool-II:

1. Se descarga el archivo fuente de la página oficial de Pgpool-II:

   http://pgfoundry.org/projects/pgpool

2. Ahora se coloca el archivo en:

   #cd /downloads/scripts_cluster/pgpool

3. Descomprimir el archivo.tar con:

   #tar xzvf pgpool-II-2.2.2.tar.gz

4. Ver los archivos de instalación README e INSTALL para conocer de manera general como realizar la instalación.

5. En la misma ruta, ejecutamos el siguiente comando:

   #./configure

6. Después ejecutamos:

   #./make

7. Y por último:

   #./make install

8. Ahora nos dirigimos a la ruta donde se encuentran los archivos de configuración de Pgpool-II, ubicados en:

   #cd /usr/local/etc

9. Ahí estarán los archivos de ejemplo para configurar el Pgpool-II que son

   pcp.conf.sample
   pgpool.conf.sample
   pool_hba.conf.sample

10. Lo que prosigue es hacer copia y renombrar dichos archivos para poder modificarlos con nuestra configuración específica.

    #cp pcp.conf.sample pcp.conf
    #cp pgpool.conf.sample pgpool.conf
    #cp pool_hba.conf.sample pool_hba.conf

11. En el archivo pcp.conf, es donde se configura el usuario y la contraseña para entrar a Pgpool-II, la contraseña se debe indicar en modo de autentificación **md5,** lo cual se obtiene tecleando lo siguiente:

    #pg_md5 –p

    Nos pide teclear el *password* que pondremos a Pgpool-II y nos lo regresará en modo de autentificación md5.

    Ejemplo: *password:<colocar password>*
    Resultado: *e8a48653851e28c69d0506508fb27fc5*

12. Con la contraseña creada se coloca en el archivo de pcp.conf, con el siguiente orden:

    usuario:password
    Ejemplo: pguser:*e8a48653851e28c69d0506508fb27fc5*

13. El segundo archivo a modificar es el de pgpool.conf, donde se configuran los parámetros necesarios para que Pgpool-II pueda realizar el proceso de *failover*.

    Aquí se muestran los principales parámetros que se modificaron dentro del archivo:

```
# pgpool-II configuration file
# Host name or IP address to listen on: '*' for all, '' for no TCP/IP
# connections
listen_addresses = '*'

# Port number for pgpool
port = 9999

# Port number for pgpool communication manager
pcp_port = 9898

# pgpool communication manager timeout. 0 means no timeout, but strongly not
recommended!
pcp_timeout = 10

# number of pre-forked child process
num_init_children = 32

# Number of connection pools allowed for a child process
max_pool = 4

# If idle for this many seconds, child exits.  0 means no timeout.
child_life_time = 300

# If idle for this many seconds, connection to PostgreSQL closes.
# 0 means no timeout.
connection_life_time = 0

# If child_max_connections connections were received, child exits.
# 0 means no exit.
child_max_connections = 0

# If client_idle_limit is n (n > 0), the client is forced to be
# disconnected whenever after n seconds idle (even inside an explicit
# transactions!)
# 0 means no disconnect.
client_idle_limit = 0

# Maximum time in seconds to complete client authentication.
# 0 means no timeout.
authentication_timeout = 60

# Logging directory
logdir = '/tmp'

# pid file name
pid_file_name = '/var/run/pgpool/pgpool.pid'

# If true, operate in master/slave mode.
master_slave_mode = true

# If true, cache connection pool.
connection_cache = true

# Health check user
health_check_user = 'postgres'

# Execute command by failover.
# special values:  %d = node id
#                  %h = host name
#                  %p = port number
#                  %D = database cluster path
#                  %m = new master node id
```

```
#                  %M = old master node id
#                  %% = '%' character
#
failover_command = '/usr/local/pgsql/bin/pgpool-failover %d %h %p %D %m %M'

# Execute command by failback.
# special values:  %d = node id
#                  %h = host name
#                  %p = port number
#                  %D = database cluster path
#                  %m = new master node id
#                  %M = old master node id
#                  %% = '%' character
#
failback_command = '/usr/local/pgsql/bin/pgpool-failback %d %h %p %D %m %M'
# system DB info
system_db_hostname = '132.248.37.185'
system_db_port = 5432
system_db_dbname = 'dbgedgapa'
system_db_schema = 'pgpool_catalog'
system_db_user = 'postgres'
system_db_password = 'xxxxx'

# backend_hostname, backend_port, backend_weight
# here are examples
# backend_hostname, backend_port, backend_weight
backend_hostname0 = '132.248.37.185'
backend_port0 = 5432
backend_weight0 = 1
backend_data_directory0 = '/usr/local/pgsql/data'
backend_hostname1 = '132.248.37.214'
backend_port1 = 5432
backend_weight1 = 1
backend_data_directory1 = '/usr/local/pgsql/data2'
backend_hostname2 = '132.248.37.38'
backend_port2 = 5432
backend_weight2 = 1
backend_data_directory2 = '/usr/local/pgsql/data'
```

En esta última parte del archivo pgpool.conf , es donde se específica cual es el orden de importancia de los nodos, el nodo maestro como es el nodo principal será configurado como el backend_hostname0 , cuando este deje de funcionar el segundo nodo en importancia será el backend_hostname1,así sucesivamente, de esta manera se configura el *failover* dentro de Pgpool-II.

14. Además de colocar en el archivo pgpool.conf los parámetros necesarios para la realización de un proceso de *failover* es necesario crear un *script* que ejecute el cambio de rol de los nodos. El script se crea en la siguiente ruta:

    #cd /usr/local/pgsql/bin

    Con el siguiente comando:

    #mkdir pgpool-failover

    El cual contiene lo siguiente:

```
#! /bin/sh
LOGGER="/usr/bin/logger -i -p local0.info -t pgpool"
BASENAME=`/usr/bin/basename $0`
ID=`/usr/bin/id -un`

# $1 = node id
# $2 = host name
# $3 = port number
# $4 = database cluster path
# $5 = new master node id
# $6 = old master node id

$LOGGER "Executing $BASENAME as user $ID"
$LOGGER "Failover of node $1 at hostname $2. New master node is $5. Old master
node was $6."
################################################################################
#Variables de ambiente tomadas por slony
################################################################################
export CLUSTERNAME=gedgapa
export MASTERDBNAME=dbprueba1
export MASTERHOST=132.248.37.214
export REPLICATIONUSER=postgres
export PGBENCHUSER=pgbench
export SLAVEDBNAME1=dbprueba2
export SLAVEHOST1=132.248.37.214
export SLAVEDBNAME2=dbprueba3
export SLAVEHOST2=132.248.37.18
################################################################################

/usr/local/pgsql/bin/slonik <<_EOF_

cluster name = $CLUSTERNAME;

node 1 admin conninfo = 'dbname=$MASTERDBNAME host=$MASTERHOST
user=$REPLICATIONUSER';
node 2 admin conninfo = 'dbname=$SLAVEDBNAME1 host=$SLAVEHOST1
user=$REPLICATIONUSER';
node 3 admin conninfo = 'dbname=$SLAVEDBNAME2 host=$SLAVEHOST2
user=$REPLICATIONUSER';


failover (id=1, backup node =2);
drop node (id=1,event node=2);

lock set (id=1, origin=1);
wait for event (origin=1,confirmed=2);
move set (id=1, old origin=1, new origin=2);
wait for event (origin=1,confirmed=2);

lock set (id=2, origin=1);
wait for event (origin=1,confirmed=2);
move set (id=2, old origin=1, new origin=2);
wait for event (origin=1,confirmed=2);

lock set (id=3, origin=1);
wait for event (origin=1,confirmed=2);
move set (id=3, old origin=1, new origin=2);
wait for event (origin=1,confirmed=2);

lock set (id=4, origin=1);
wait for event (origin=1,confirmed=2);
move set (id=4, old origin=1, new origin=2);
wait for event (origin=1,confirmed=2);

lock set (id=5, origin=1);
```

```
wait for event (origin=1,confirmed=2);
move set (id=5, old origin=1, new origin=2);
wait for event (origin=1,confirmed=2);

_EOF_

exit 0
```

En el script se indica que nodo ocupa el rol de nodo maestro, si ocurre un proceso de failover, se indica el movimiento de los sets de un nodo a otro, Aquí sólo se muestran los primeros cinco sets.

15. Es necesario crear el directorio pgpool en:

    #cd /var/run

    Con el siguiente comando:
    #mkdir pgpool

    Se crea para almacenar ahí los archivos *logs* del proceso desde que arranca.

16. Ahora se inicia el servicio de Pgpool-II, con el *script* 1start_pgpool, ubicado en

    #cd /downloads/scripts_cluster/pgpool

    Para levantar el servicio se ejecuta:

    #./1start_pgpool.sh

```
#!/bin/sh
#############################################################################
pgpool -c -f /usr/local/etc/pgpool.conf -a /usr/local/etc/pool_hba.conf -
F /usr/local/etc/pcp.conf -n -d
```

17. Ahora si se desea reiniciar o parar el servicio se ejecuta el siguiente *script*:

    #./2stop_pgpool.sh

```
#!/bin/sh
#############################################################################
pgpool -f /usr/local/etc/pgpool.conf -F /usr/local/etc/pcp.conf -m f stop
```

Instalación de la consola de PgpoolAdmin 2.2

Con la instalación de Pgpool-II, se tiene la funcionalidad de *failover* dentro del *cluster*, ahora lo que se desea tener es una herramienta más práctica para manejar dicha funcionalidad, por lo tanto, se instalará PgpoolAdmin 2.2, que será la interfaz gráfica para manejar la administración de Pgpool-II.

A continuación se listarán los pasos necesarios para la instalación de PgpoolAdmin 2.2:

1. Se descarga el archivo fuente de la página oficial de Pgpool-II:

   http://pgfoundry.org/projects/pgpool

2. Ahora se coloca el archivo en:

   #cd /downloads/scripts_cluster/pgpool

3. Descomprimir el archivo.tar con:

   #tar xzvf pgpoolAdmin.2.2.tar.gz

4. Ahora es necesario mover los archivos de pgpoolAdmin a la siguiente ubicación:

   #cd /var/www/html

   Con el siguiente comando se mueven a dicha ruta:

   #mv pgpoolAdmin.2.2 / var/www/html

5. Ahora se podrá visualizar en un explorador de internet la consola de administración de Pgpool, donde se observarán el estado de los nodos del *cluster* como se muestra en la figura 4.14 :

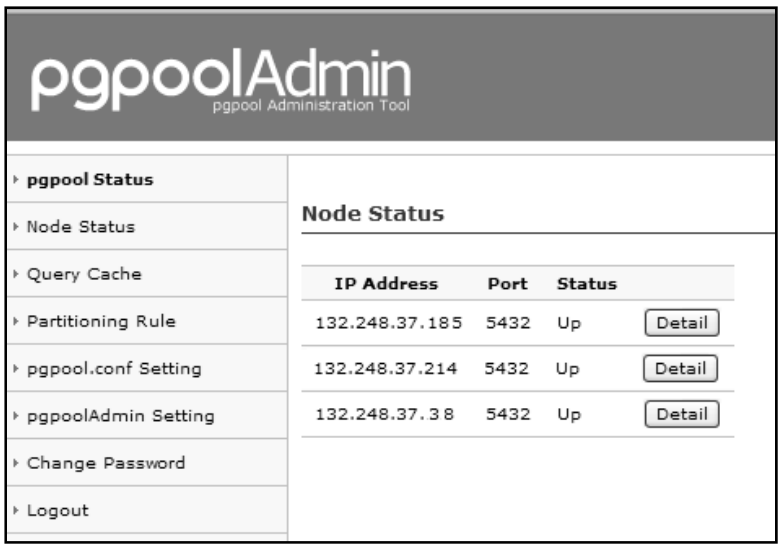


Figura 4.14 Estado de los nodos del cluster.

En la figura 4.15 se muestra la ventana de opciones de configuración de Pgpool.

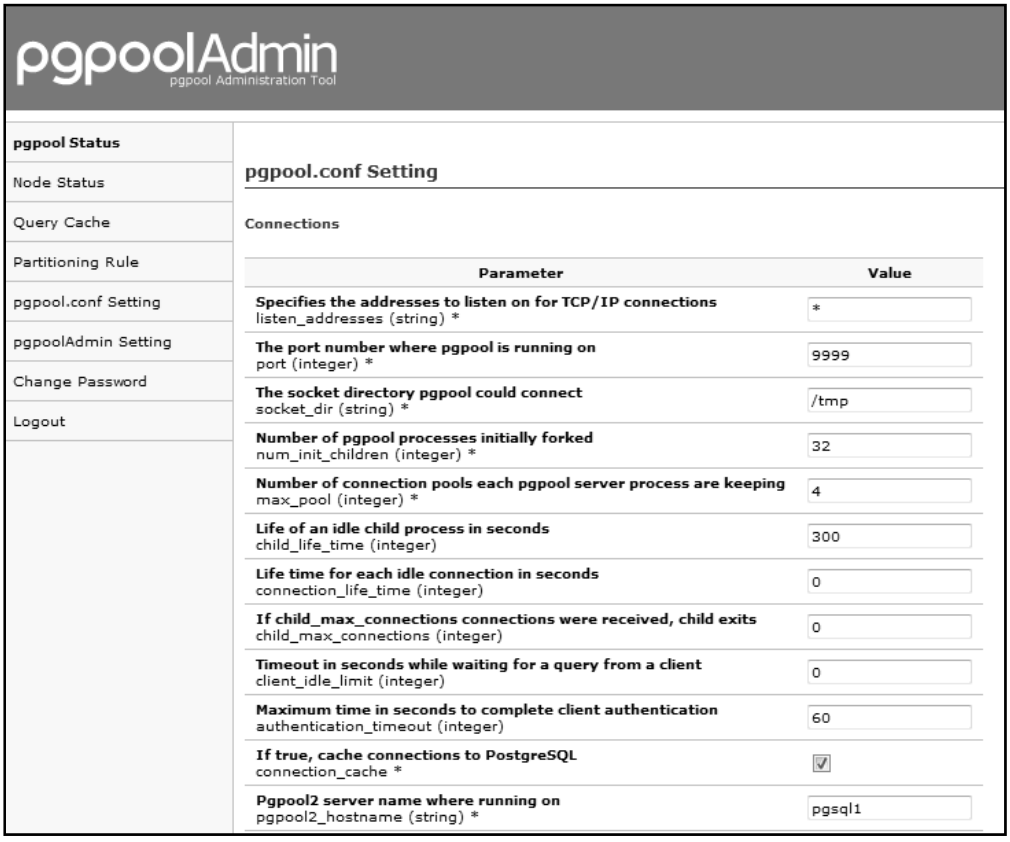


Figura 4.15 Ventana de configuración de Pgpool.

4.2.9 Pruebas

En este subcapítulo se explican las pruebas que se realizaron para comprobar que el proceso de replicación se efectúe adecuadamente, así como también, el proceso de *failover*, cambiando un nodo esclavo al lugar de un nodo maestro en presencia de alguna falla en el nodo maestro original.

Pruebas de replicación

Con la puesta en marcha de Slony-I dentro del *cluster* se cumple con la replicación, ahora es necesario comprobar que dicha replicación funcione correctamente, por dicho motivo es necesario poner el sistema de replicación a prueba y ver si los resultados satisfacen las necesidades planteadas inicialmente.

Poner a prueba Slony-I significa insertar, borrar o actualizar registros de las tablas que componen a dbgedgapa y observar dichas operaciones en los nodos esclavos.

A continuación se enlistará la secuencia de pasos de la realización de la pruebas de replicación:

1. Se entra al esquema en el cual se desea modificar una tabla, en este caso es el esquema admin y la tabla es cgrupos, en la figura 4.16 se observa la cantidad de registros que contiene originalmente:

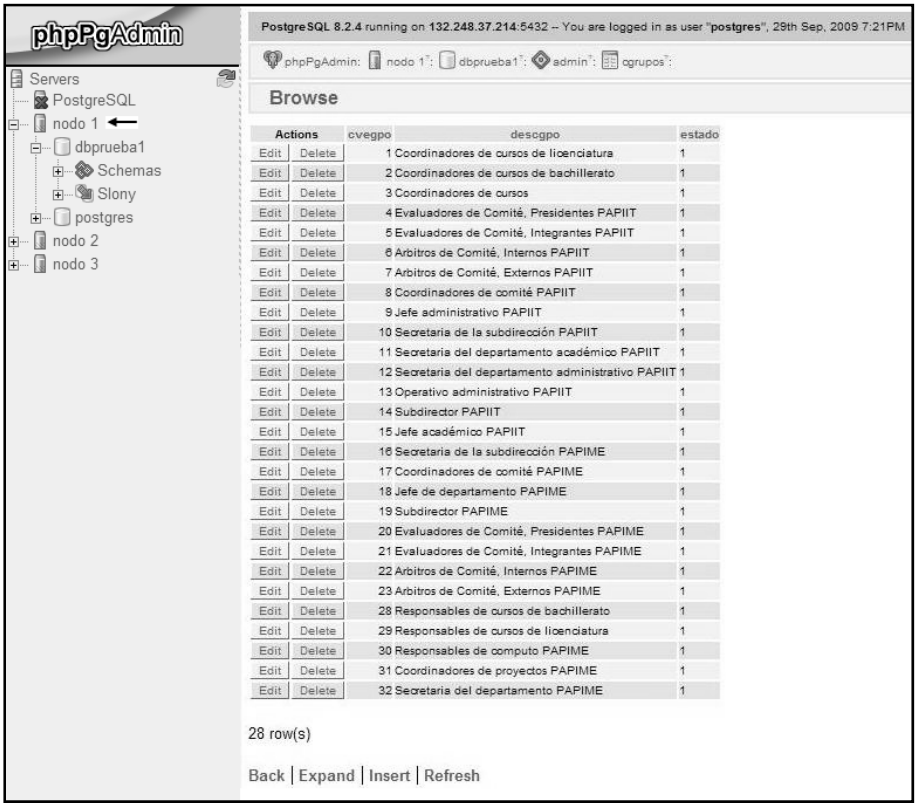


Figura 4.16 Vista de la tabla original cgrupos dentro del nodo maestro.

2. Ahora se inserta un nuevo registro en la tabla cgrupos (figura 4.17).

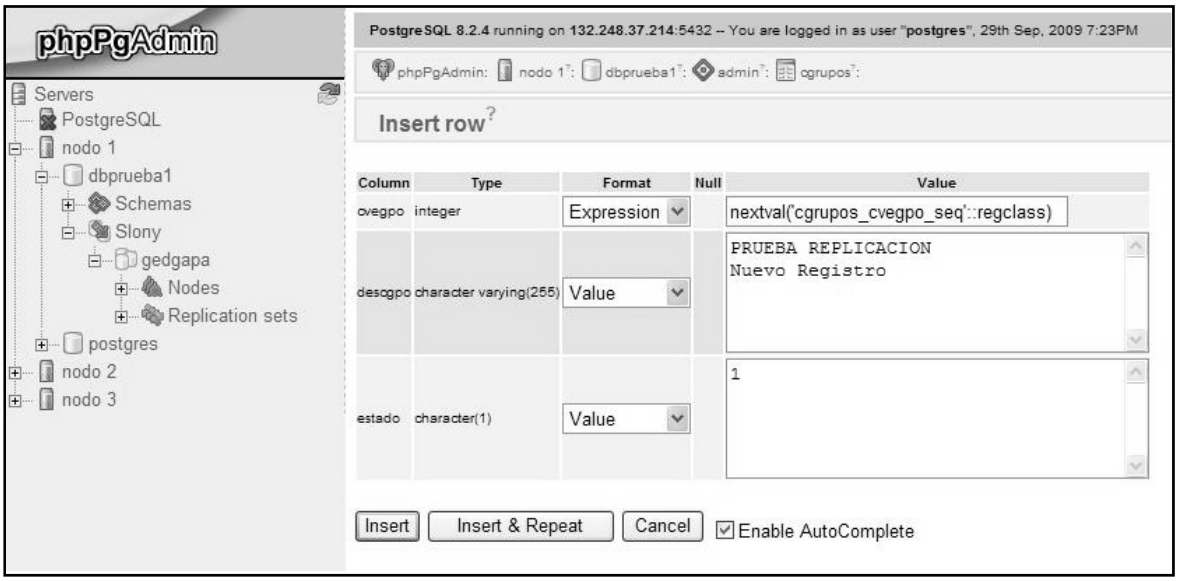


Figura 4.17 Insertando un nuevo registro dentro de la tabla.

3. Se observa que el registro se haya insertado correctamente dentro de cgrupos (figura 4.18).

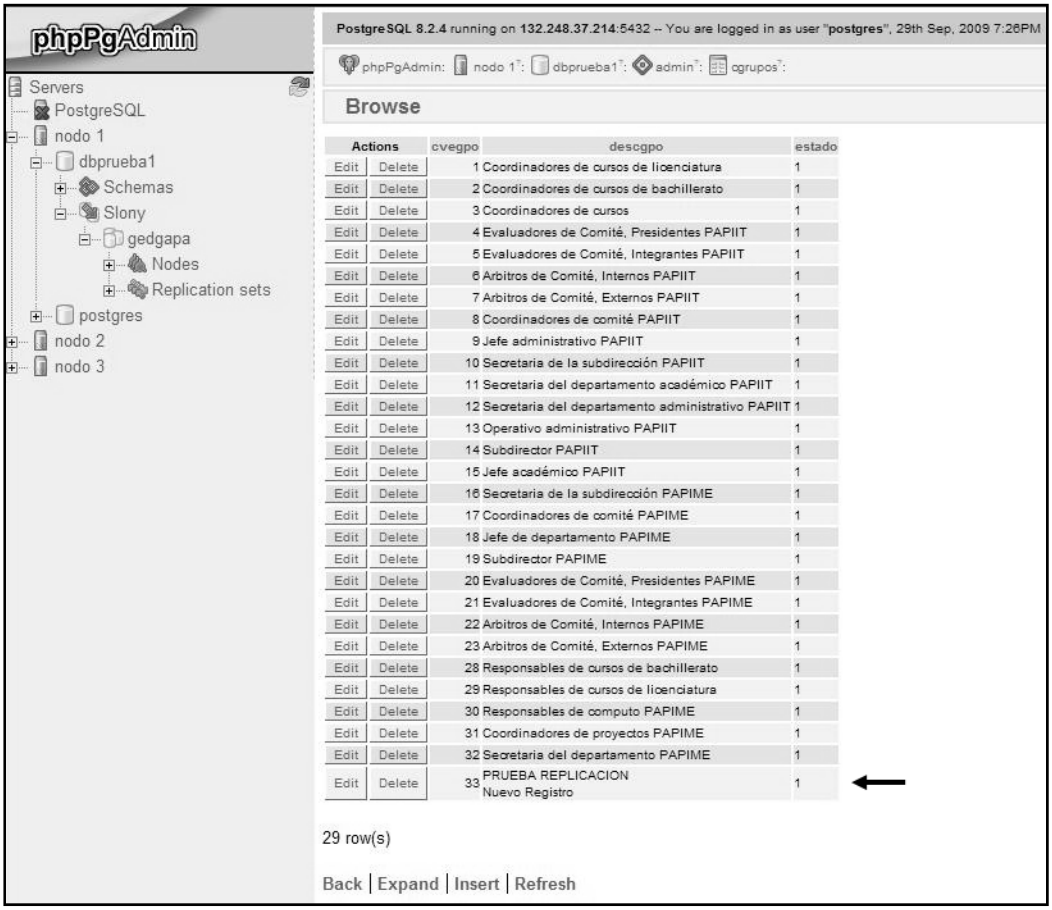


Figura 4.18 Mostrando la correcta inserción del nuevo registro en el nodo maestro.

4. Ahora se observa que el nuevo registro replicado en el nodo esclavo 1 (figura 4.19).

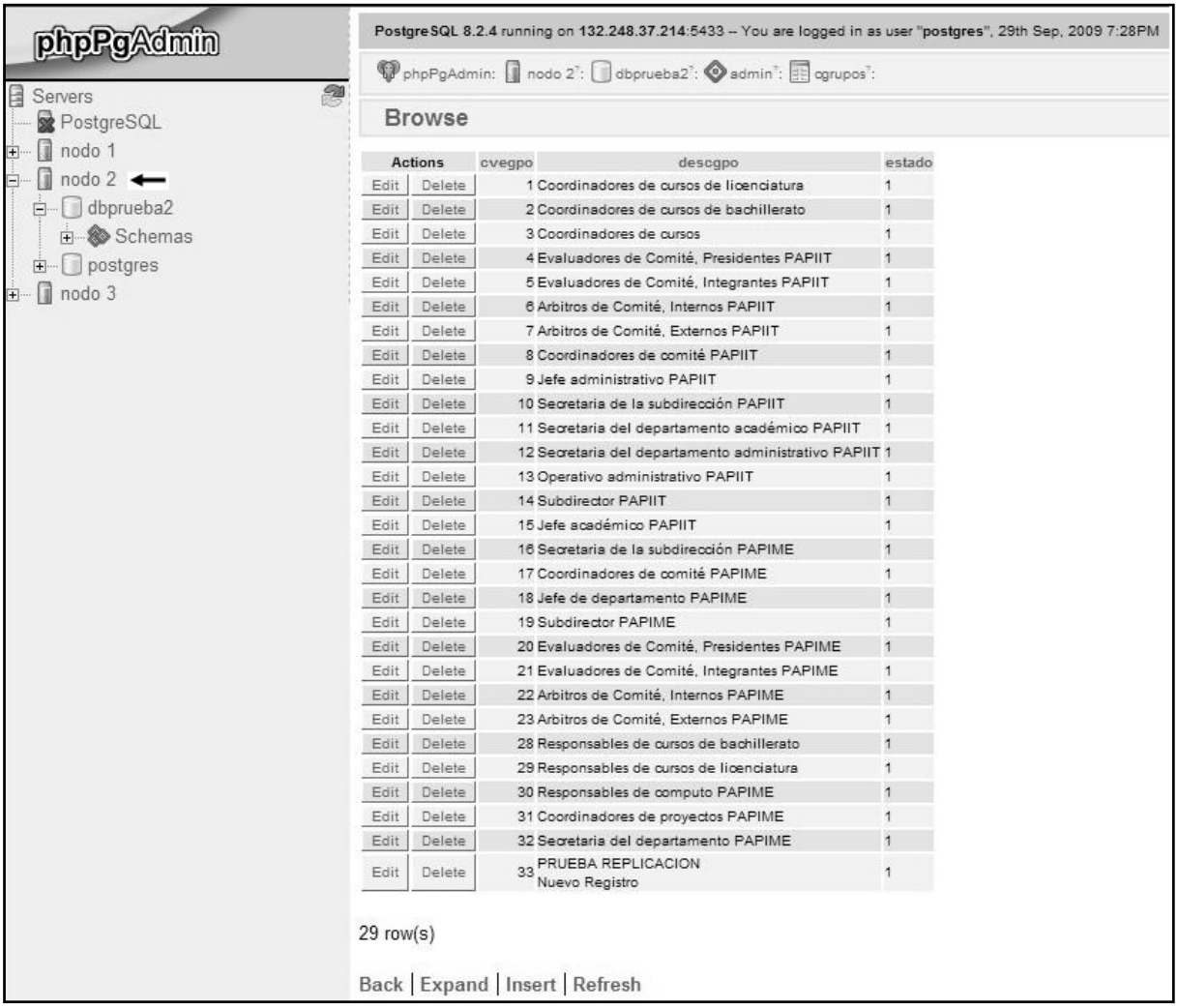


Figura 4.19 Mostrando el nuevo registro replicado en el nodo esclavo 1.

5. También se observa el nuevo registro replicado en el nodo esclavo 2 (figura 4.20).

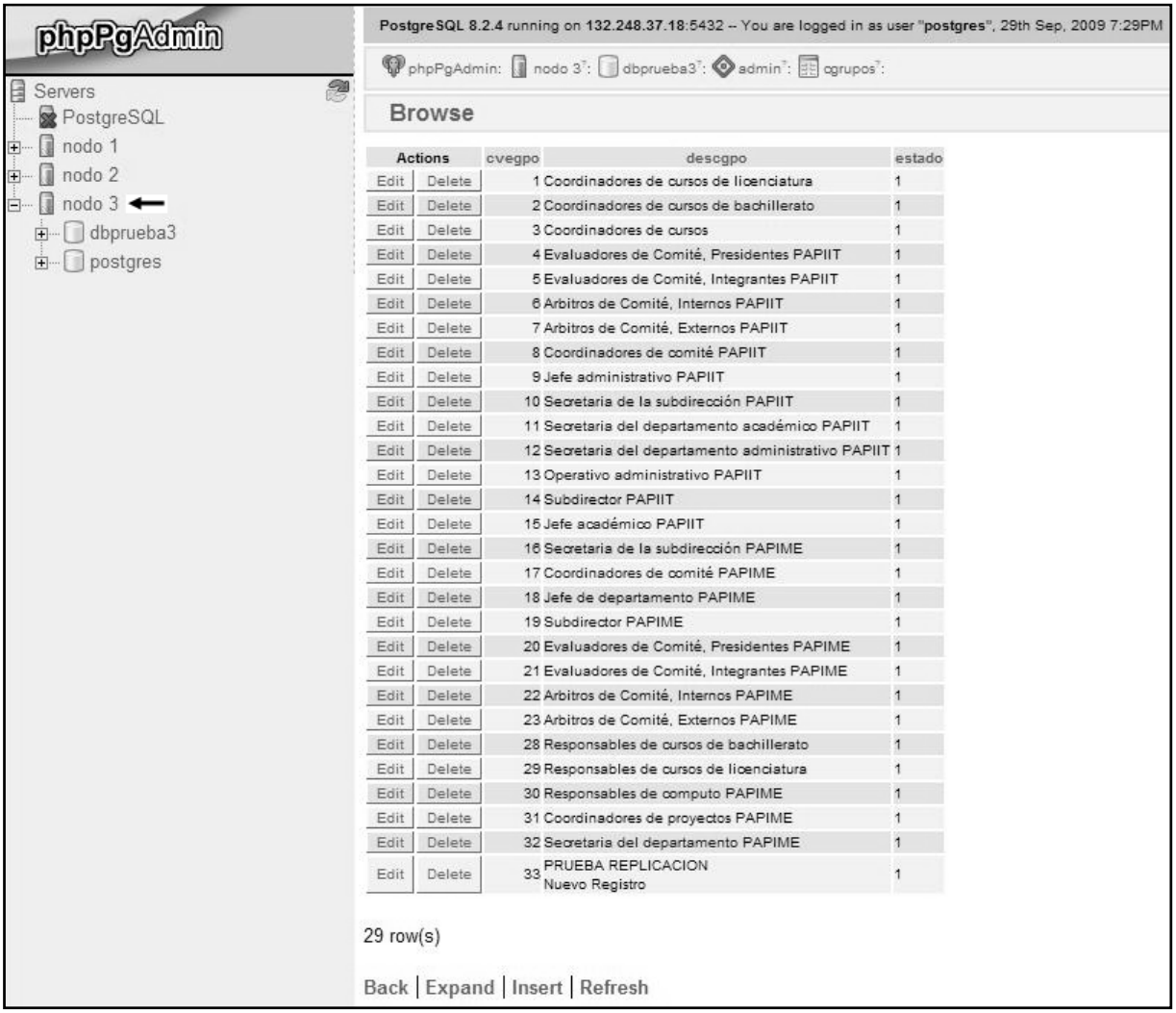


Figura 4.20 Mostrando el nuevo registro replicado en el nodo esclavo 2.

Con esta simple prueba se comprueba que el motor de Slony-I está replicando correctamente en ambos nodos esclavos.

Prueba de Failover

Con la realización de las pruebas de replicación, es conveniente poner a prueba el sistema de replicación provocando un fallo de operación en el nodo maestro, matando al nodo y forzando a un proceso de *failover* entre los nodos, a continuación se observa los resultados:

1. El primer paso es verificar el estado de cada uno de los nodos, podemos checarlo desde la consola de phpPgAdmin (figura 4.21) o desde la consola de PgpoolAdmin (figura 4.22):

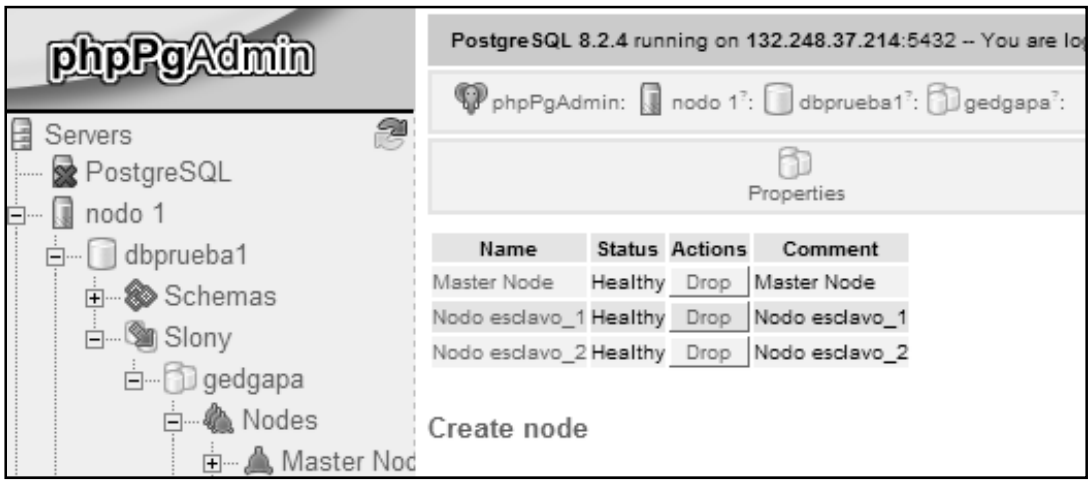


Figura 4.21 Estado de los nodos desde phpPgAdmin.

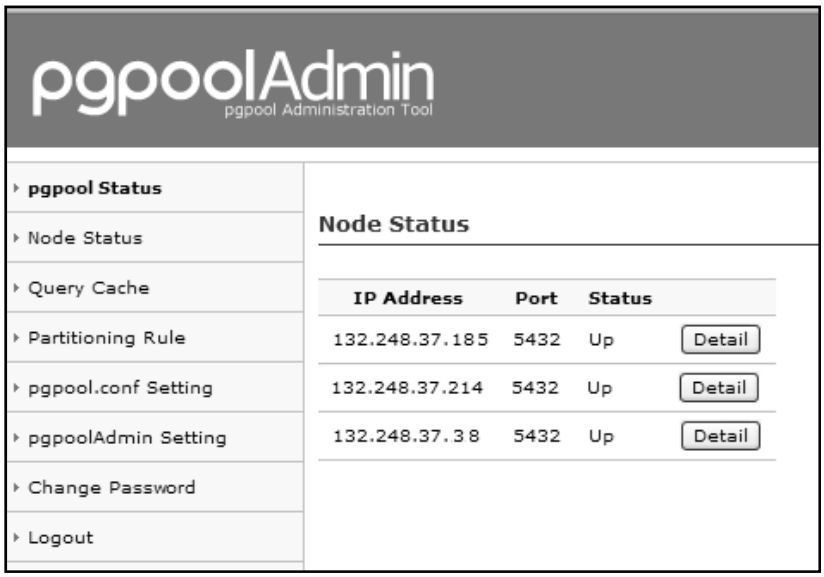


Figura 4.22 Estado de los nodos desde phpPgAdmin.

2. Para verificar que el nodo 1 , tiene el rol de nodo maestro dentro de la replicación, se tratará de insertar un nuevo registro en el nodo 2 , es decir , desde el esclavo1 y se observará que la operación no será válida porque el único que tiene permisos de escritura es el nodo 1 o nodo maestro (figura 4.23).

Figura 4.23 Error al insertar nuevo registro en el nodo 2

Donde el error menciona que la tabla es replicada y no puede ser modificada por un nodo suscriptor.

3. Ahora se procede a realizar la baja del nodo maestro para forzar un proceso de *failover*, aquí se dará la baja del nodo desde la consola, pero bien se puede apagar el equipo, cualquiera de las dos cosas pondrá al nodo maestro fuera de servicio (figura 4.24).

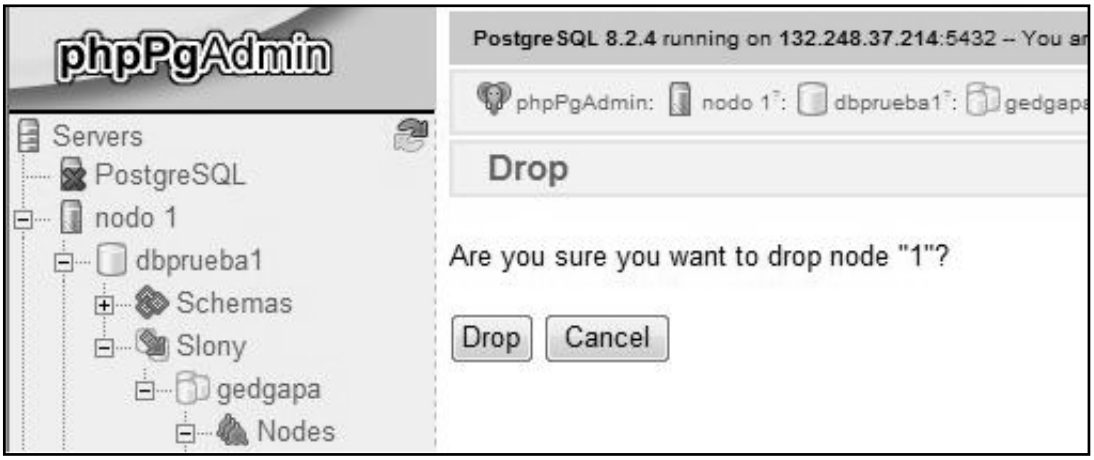


Figura 4.24 Ventana de confirmación de baja el nodo 1.

4. Checamos el estado del nodo maestro desde la consola de Pgpool (figura 4.25).

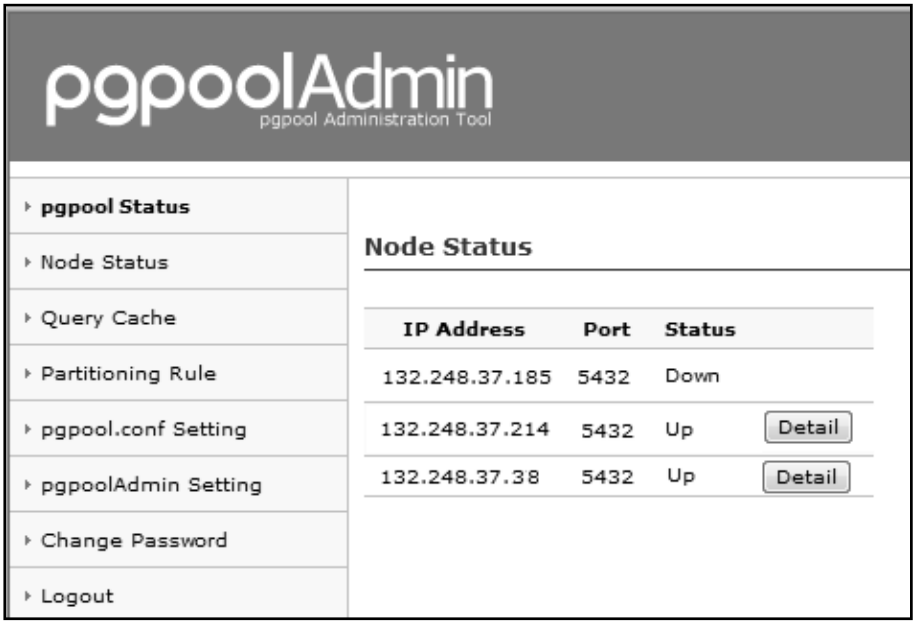


Figura 4.25 Estado de fuera de servicio del nodo maestro.

5. Ahora entrará el proceso de *failover* donde Pgpool realizará el cambio de nodo maestro para que este responda las peticiones de conexión a la base de los clientes. Entonces como se mostró previamente en el *script* de configuración pgpool.conf, el nodo a tomar el rol de nuevo nodo maestro es el nodo esclavo2. Para probar que el nodo esclavo 2 es el nuevo nodo maestro se inserta un nuevo registro, ya que ahora este nodo tiene privilegios de escritura (figura 4.26).

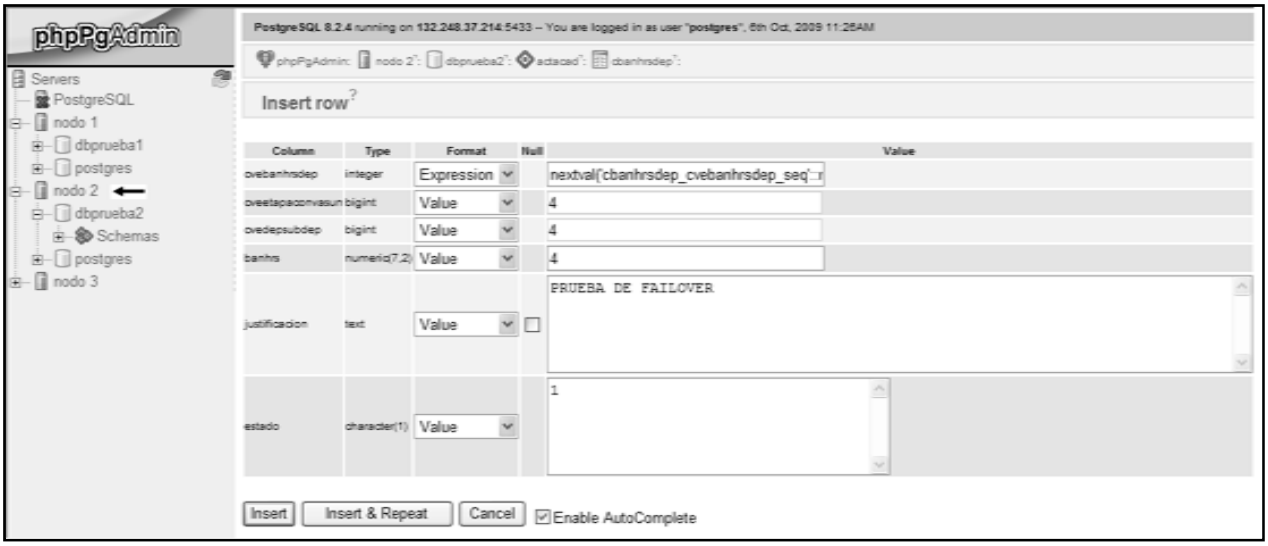


Figura 4.26 Ventana de inserción de nuevo registro en el nuevo nodo maestro (nodo esclavo 1).

6. En la siguiente ventana se aprecia cómo fue insertado el nuevo registro, confirmando que el nodo 2 tiene permiso de escritura por ser el nuevo nodo maestro.

Figura 4.27 Ventana del nuevo nodo maestro con el registro recién creado.

De manera general, así funciona el proceso de *failover* entre los nodos del *cluster*, intercambiando roles según se necesiten junto con la Pgpool que indica el orden del intercambio de los roles de los nodos.

Las pruebas de replicación y *failover* presentadas en este capítulo son resultado de varios intentos realizadas con anterioridad donde se presentaron errores que se tuvieron que investigar y corregir, a continuación en la tabla 4.1 se resumen las pruebas más representativas:

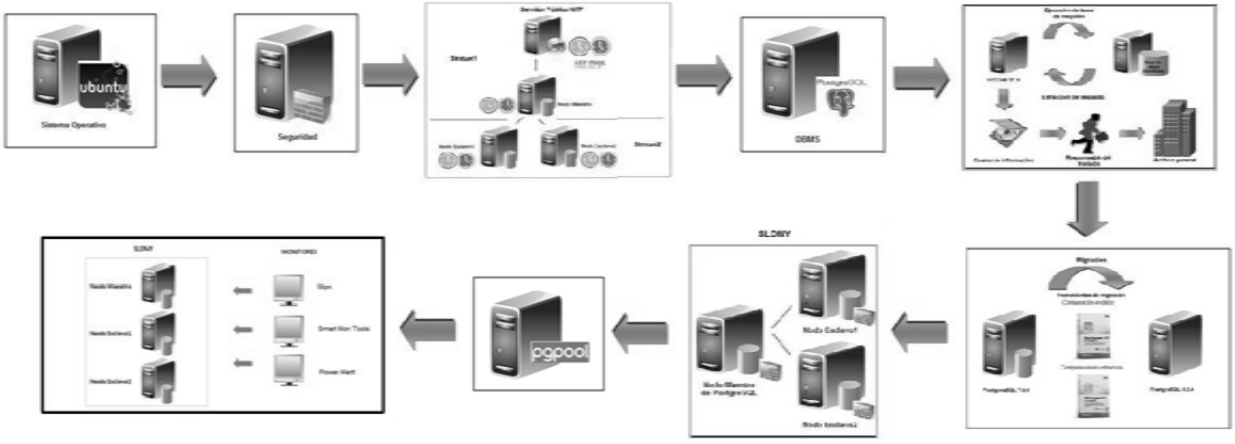
Tabla 4.1 Cronología de pruebas

Pruebas	Estado	Notas
ESCENARIO1 (2 nodos)		
Nodo maestro replicando a un esclavo	Exitoso	------
Cambiar el nodo maestro a suscriptor y el esclavo a replicador	Exitoso	------
ESCENARIO2 (3 nodos)		
Nodo maestro replicando a un esclavo	Exitoso	------
Agregar nodo esclavo 3	Fallido	No actualizaba los datos
Agregar nodo esclavo 3	Exitoso	------
Nodo maestro replicando a dos esclavos	Exitoso	------
Failover de nodo esclavo 1 a nodo maestro	Fallido	ERROR: Slony-I: set 1 does not originate on local node
Failover de nodo esclavo 1 a nodo maestro	Exitoso	Corrección en el script de configuración de slony
Failover de nodo esclavo 2 a nodo maestro	Exitoso	------

Agregar nodo 1 de nuevo a la replicación.	Exitoso	------
Cambiar nodo 1 de esclavo a nodo maestro	Exitoso	------

La realización de pruebas consiste en un proceso de retroalimentación del *cluster*, debido a que se pone a prueba el funcionamiento del mismo, buscando fallas para ser corregidas para evitar que se presenten cuando se esté en producción y el desempeño del *cluster* sea el óptimo.

4.2.10 Monitoreo



Instalación de Mon

Mon es un *software* que se configura para monitorear los nodos del *cluster* o cualquier otro dispositivo que se encuentre conectado a la red, que tenga una función en particular que se deba monitorear, con la ventaja de que si llegará a fallar dicho dispositivo Mon mandará una alerta vía e-mail al administrador de la red, para notificarle que dicho servicio se encuentra no disponible.

A continuación se listarán los pasos necesarios para instalar Mon en un ambiente Linux:

1. Abrir una terminal.

2. Entrar como root y teclear la contraseña

#sudo -i

3. Para instalar la herramienta MON, tecleamos:

#apt-get install mon

Nota: Es necesario verificar que se tengan todos los repositorios de Ubuntu instalados para para que se realice la instalación de mon.

4. Instalar Perl, Mon necesita de un módulo proveniente de Perl, por ello es necesario instalarlo previamente con:

#apt-get install perl

5. MON necesita 3 módulos indispensables para su funcionamiento y se instalan en el directorio de Mon:

#cd /etc/mon

Dichos módulos son

CPAN
#apt-get install check cpan
Time::Period
#apt-get install check period
Time::HiRes
#apt-get install check time

6. En el monitoreo de los nodos se utiliza el mail.alert que brinda el servicio de enviar un e-mail al administrador de la red, para informarle de la falla en alguno de los servicios monitoreados, para que se tomen las medidas pertinentes. Con el siguiente comando instalamos mail.alert:

#apt-get install sendmail

8. Mon maneja dos archivos de configuración:

auth.cf - Archivo de autentificación de usuarios.
mon.cf – Archivo de configuración del monitoreo de los servicios.

Ubicados en:

#cd /etc/mon

9. Ahora se configurará el archivo auth.cf donde se deben colocar los usuarios que tendrán acceso a Mon, para nuestro caso será el administrador de la red y *root* con sus respectivas contraseñas generadas con el comando htpasswd.

El archivo auth.cf es el que se muestra a continuación:

```
#source_host      user  password
#
# allow from user "mon" from any host
#
# * mon monpassword
#
# allow from host 127.0.0.1 without requiring
# a valid username and password
#
```

```
127.0.0.1   root  B5YOcQq1naG3E
132.248.37.10   administrador B5YOcQq1naG3E
#
```

10. Ahora se procede a configurar los servicios que se desean monitorear en el archivo mon.cf, como se muestra a continuación:

```
#
# group definitions (hostnames or IP addresses)
#

hostgroup nodo1 132.248.37.185
hostgroup nodo2 132.248.37.214
hostgroup nodo3 132.248.37.32
#
# For the servers in building 1, monitor ping and telnet
# BOFH is on weekend call :)
#
watch nodo1
   service ping
      description ping al nodo1
      interval 1m
      monitor fping.monitor
      allow_empty_group
      period wd {Sun-Sat}
      alert mail.alert
      felipe@dgapa.unam.mx,alejo@dgapa.unam.mx,bahena@dgapa.unam.mx
      alertevery 24h
   service postgrestcp
      description ping al servidor de postgres de base de datos
      interval 1m
      monitor tcp.monitor -p 5432
      period wd {Sun-Sat}
      alert mail.alert
      felipe@dgapa.unam.mx,alejo@dgapa.unam.mx,bahena@dgapa.unam.mx
      alertevery 10m
service http
      description servicio de apache en nodo1
       interval 1m
       monitor http.monitor
       allow_empty_group
       period wd {Sun-Sat}
       alert mail.alert
        felipe@dgapa.unam.mx,alejo@dgapa.unam.mx,bahena@dgapa.unam.mx
       alertevery 24h



watch nodo2
   service ping
       description ping al nodo1
      interval 1m
       monitor fping.monitor
       allow_empty_group
       period wd {Sun-Sat}
       alert mail.alert
      felipe@dgapa.unam.mx,alejo@dgapa.unam.mx,bahena@dgapa.unam.mx
       alertevery 24h
   service postgrestcp
      description ping al servidor de postgres de base de datos en el nodo2
      interval 1m
      monitor tcp.monitor -p 5432
      period wd {Sun-Sat}
```

```
      alert mail.alert
      felipe@dgapa.unam.mx,alejo@dgapa.unam.mx,bahena@dgapa.unam.mx
      alertevery 10m


watch nodo3
   service ping
      description ping al nodo3
      interval 1m
      monitor fping.monitor
      allow_empty_group
      period wd {Sun-Sat}
      alert mail.alert
      felipe@dgapa.unam.mx,alejo@dgapa.unam.mx,bahena@dgapa.unam.mx
      alertevery 24h
   service postgrestcp
      description ping al servidor de postgres de base de datos en el nodo3
      interval 1m
      monitor tcp.monitor -p 5432
      period wd {Sun-Sat}
      alert mail.alert
      felipe@dgapa.unam.mx,alejo@dgapa.unam.mx,bahena@dgapa.unam.mx
      alertevery 10m
```

Donde se indica que servicios que se desean configurar, con que intervalo de tiempo y los *e-mails* de los administradores cuando se necesario mandar una alerta.

11. La consola de Mon se encuentra en:

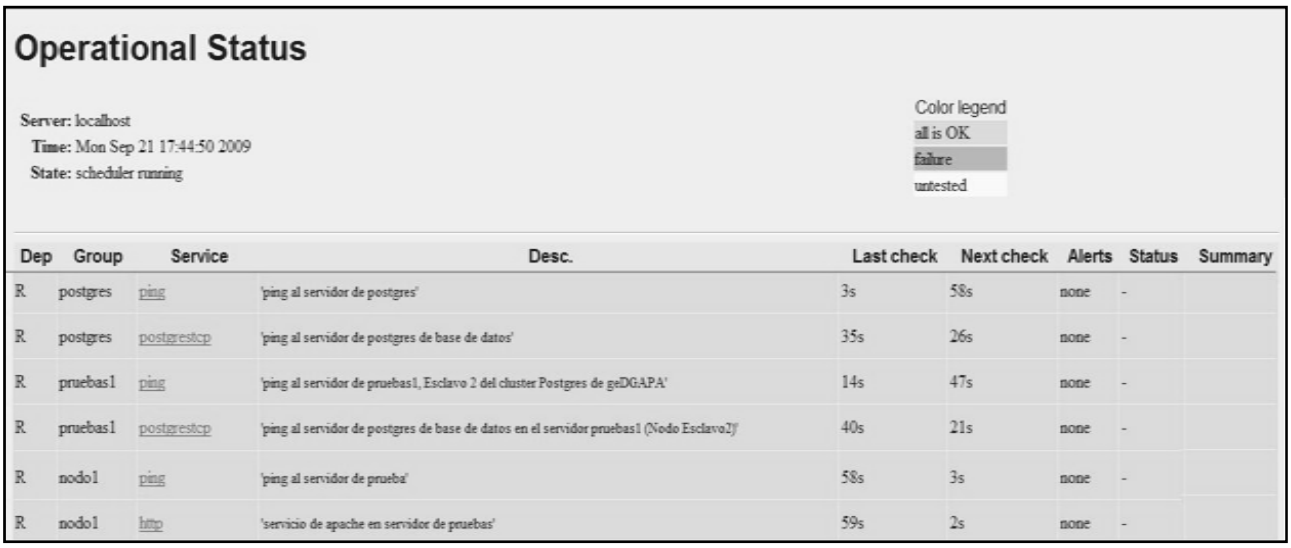
#cd /usr/lib/cgi-bin/mon.cgi

Nota: Para poder observar la consola en un explorador de internet es necesario agregar el archivo mon.cgi al cgi.bin del servidor web.

12. Ahora desde la siguiente dirección se podrá observar la interfaz gráfica de la consola de Mon:

http://localhost/cgi-bin/monshow

En la figura 4.28 se muestra la consola de monitoreo de Mon y se aprecia el estado en que se encuentra cada servicio.



Operational Status

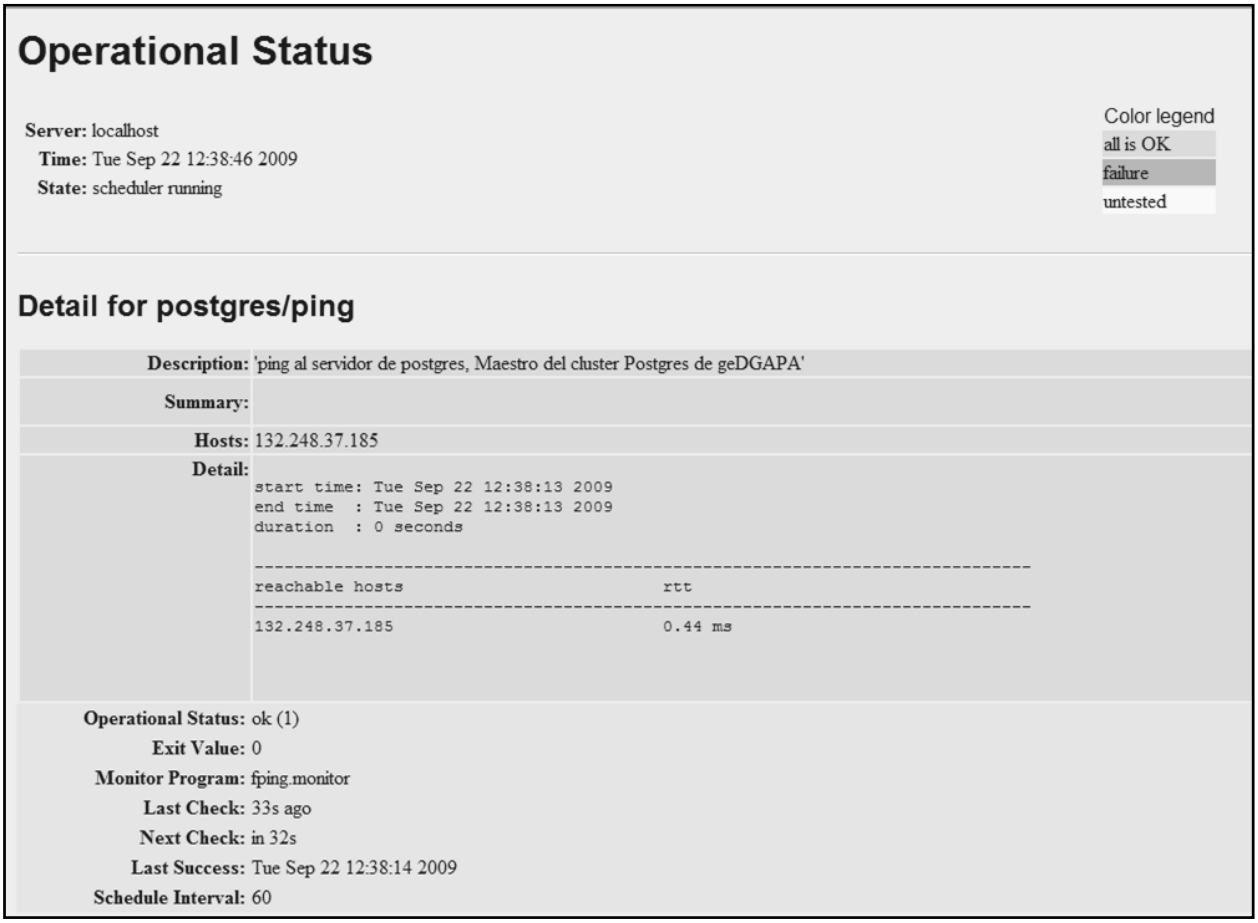
Server: localhost
Time: Mon Sep 21 17:44:50 2009
State: scheduler running

Color legend
all is OK
failure
untested

Dep	Group	Service	Desc.	Last check	Next check	Alerts	Status	Summary
R	postgres	ping	'ping al servidor de postgres'	3s	58s	none	-	
R	postgres	postgrestcp	'ping al servidor de postgres de base de datos'	35s	26s	none	-	
R	pruebas1	ping	'ping al servidor de pruebas1, Esclavo 2 del cluster Postgres de geDGAPA'	14s	47s	none	-	
R	pruebas1	postgrestcp	'ping al servidor de postgres de base de datos en el servidor pruebas1 (Nodo Esclavo2)'	40s	21s	none	-	
R	nodo1	ping	'ping al servidor de prueba'	58s	3s	none	-	
R	nodo1	http	'servicio de apache en servidor de pruebas'	59s	2s	none	-	

Figura 4.28 Consola del estado de los servicios.

En la figura 4.29 se muestra a detalle el estado del nodo maestro, el estado es *OK*, es decir, que se encuentra saludable, además se observa también cuando se realizó el último chequeo y en cuantos segundos se realizará el próximo.



Operational Status

Server: localhost
Time: Tue Sep 22 12:38:46 2009
State: scheduler running

Color legend
all is OK
failure
untested

Detail for postgres/ping

Description: 'ping al servidor de postgres, Maestro del cluster Postgres de geDGAPA'
Summary:
Hosts: 132.248.37.185
Detail:

```
start time: Tue Sep 22 12:38:13 2009
end time  : Tue Sep 22 12:38:13 2009
duration  : 0 seconds

-------------------------------------------------------------------------------
reachable hosts                          rtt
-------------------------------------------------------------------------------
132.248.37.185                           0.44 ms
```

Operational Status: ok (1)
Exit Value: 0
Monitor Program: fping.monitor
Last Check: 33s ago
Next Check: in 32s
Last Success: Tue Sep 22 12:38:14 2009
Schedule Interval: 60

Figura 4.29 Ventana de detalle del servicio.

En la figura 4.30 se muestra nuevamente a detalle el estado del nodo maestro, pero a diferencia de la figura anterior, el estado que muestra es *failure*, es decir, de falla en ese momento el demonio de Mon mandará un e-mail al administrador.

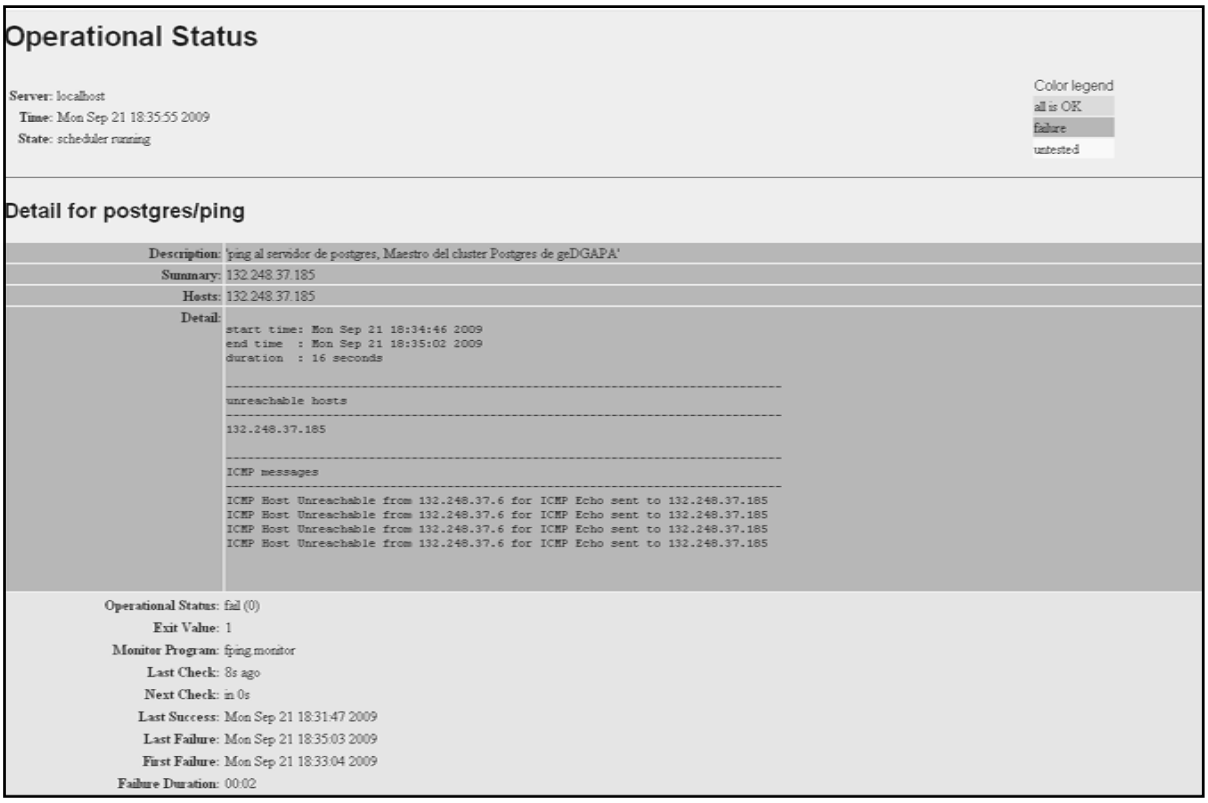


Figura 4.30 Ventana de detalle de un servicio fallando.

Instalación de Power Alert

PowerAlert permite supervisar y controlar un solo sistema de **UPS** o sistemas múltiples de UPS conectados a un pc vía **USB** o cableado de comunicaciones, durante una pérdida de energía, PowerAlert detecta que el UPS está funcionando con la energía de su batería y ordena salvar cualquier archivo abierto, evitando pérdida de datos y errores en el sistema operativo. El tiempo en el que se deben apagar los UPS se configura según se solicite.

PowerAlert también actúa como un proxy de SNMP, permitiendo que el UPS aparezca en la red local TCP/IP como dispositivo SNMP monitoreable y controlable a través del sistema de gestión de la red de PowerAlert.

Entonces, Power Alert funcionará cuando se interrumpa el suministro de energía en los nodos y equipos que componen el *cluster*, mandando a apagarlos de manera adecuada, es decir, no bruscamente para evitar algún daño ya sea físico en algún componente del equipo o de *software*.

A continuación se listarán los pasos necesarios para instalar Power Alert en un ambiente Windows:

1. Conectar el UPS a un puerto USB del equipo.

2. Instalar en el equipo Java Runtime Environment, la versión más actual y las instrucciones de instalación se encuentran en:

www.java.com

3. La instalación de Power se puede realizar de dos maneras, la primera con el disco de instalación, sí es que se cuenta con él o de lo contrario se descarga de la siguiente página:

http://www.tripplite.com/software

4. Seguir los pasos del instalador de Power Alert.

5. Después de la instalación del *software* de Power Alert automáticamente detectará el UPS.

6. Ahora se abrirá la ventana de configuración de los eventos de Power Alert (*Events Settings*), donde se configura el tiempo o la acción que debe de ejecutar el *software* en determinado evento, es decir, cuando la batería del UPS este baja o cuando la comunicación se haya perdido, entre otros eventos como se muestra en la figura 4.31:

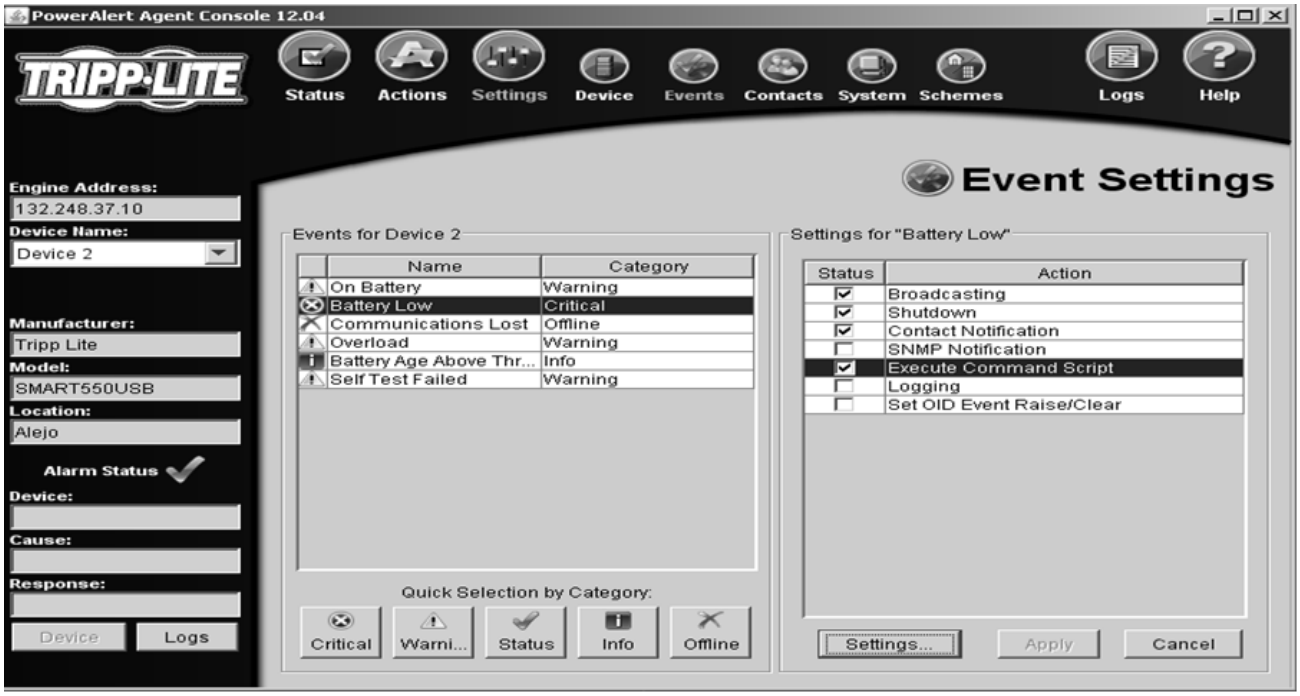


Figura 4.31 Configuración de eventos.

En la figura 4.32 se muestra, la ventana de ejecución, está opción de Power Alert permite configurar un programa o *script* que se desea ejecutar cierto tiempo después de que haya ocurrido la falla eléctrica, por ejemplo, *un script* que apague todos los servidores antes de que la batería del UPS se termine y sean apagados bruscamente.

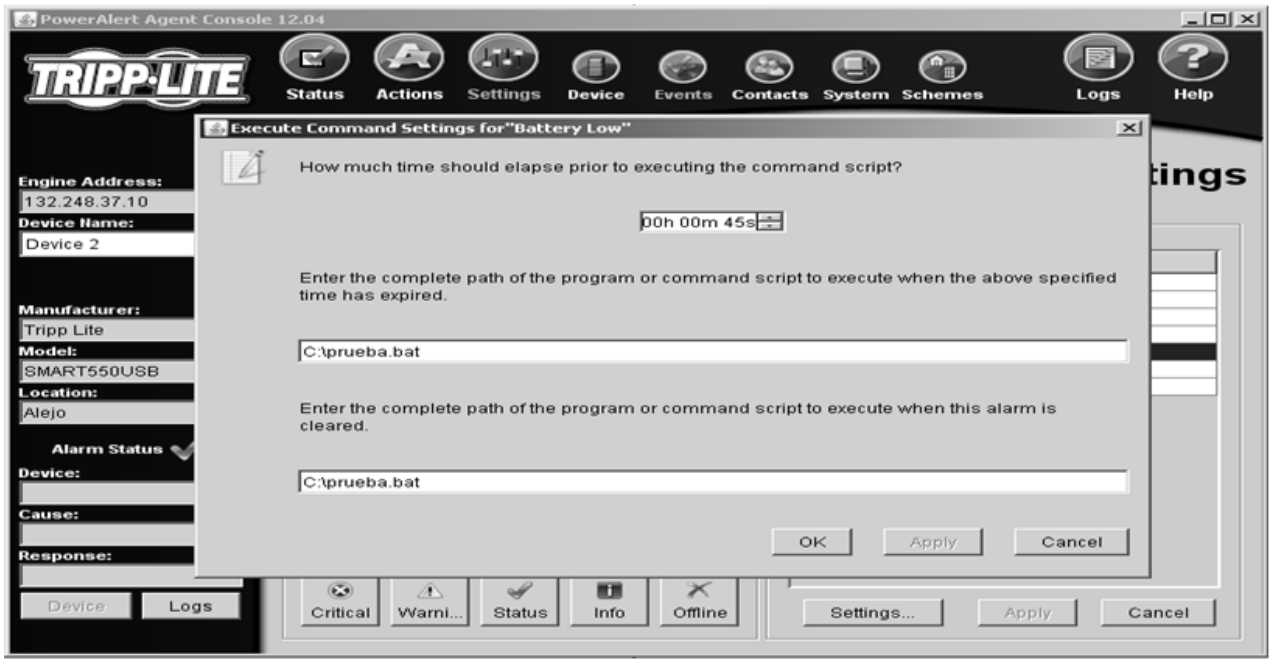


Figura 4.32 Ventana de ejecución automática.

Un ejemplo de una tarea a ejecutar sería como el contenido del siguiente archivo .bat :

```
ANT_HOME=C:\apache-ant-1.6.2
JAVA_HOME=C:\j2sdk1.4.1_01
PATH=%PATH%;%ANT_HOME%\bin
ant 1_baja_proceso_217
```

Donde por variables de ambiente se establece comunicación con **ANT** para dar de baja el proceso 217.

La figura 4.33 muestra la ventana detalle del estado de un UPS.

Figura 4.33 Ventana detalle del UPS.

Instalación de SmartMonTools

Smartmoontools es un conjunto de herramientas que nos empezarán a alertar cuando detecten que los discos físicos de los equipos que componen el *cluster* estén presentando fallas. Normalmente los discos tienen un tiempo de vida limitado, al ser un aparato mecánico las piezas dejan de girar correctamente, se llenan de polvo y dejan de funcionar, con esta herramienta tendremos estos errores controlados.

A continuación se listarán los pasos necesarios para instalar SmartMonTools en un ambiente Linux:

1. Abrir una terminal.

2. Instalar SmartMonTools con el siguiente comando:

apt-get install smartmontools

3. Nos dirigimos a la ruta donde se instaló la herramienta:

#cd /etc/default

4. Ahí abrimos con un editor de texto el archivo smartmontools con:

#vim smartmontools

5. Editamos el archivo, manteniendo comentadas todas las líneas excepto la siguiente:

start_smartd=yes

6. Ahora se configuraran los discos físicos que se tienen en el equipo, para ello se utiliza el siguiente comando:

#fdisk –l

Y muestra lo siguiente:

root@nodo1:~# fdisk -l

```
Disk /dev/sdb: 73.4 GB, 73407865856 bytes
255 heads, 63 sectors/track, 8924 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
Disk identifier: 0x000389db

Device Boot          Start         End      Blocks   Id  System
/dev/sdb1   *           1        8555    68718006   83  Linux
/dev/sdb2            8556        8924     2963992+   5  Extended
/dev/sdb5            8556        8924     2963961   82  Linux swap
/ Solaris

Disk /dev/sdc: 73.4 GB, 73407865856 bytes
255 heads, 63 sectors/track, 8924 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
Disk identifier: 0x00000000
```

Donde se observa los discos y particiones del servidor.

7. Con el conocimiento de los discos físicos con los que cuenta el equipo se configura el Smart Mon Tools con los discos que tiene que monitorear y esto se realiza en el archivo smartd.conf que se encuentra en:

#cd /etc/smartd.conf

Donde se descomentan las líneas que muestran los discos que queremos monitorear del equipo, como se muestra a continuación:

```
/dev/sdb –a –d sat –m root

/dev/sdc –a –d sat –m root
```

Las opciones –d sat es para que muestre todos los errores que salgan en los logs y –m *root* para que mande todo al e-mail de *root*, que será el mail del administrador, por lo tanto, se tendrá conocimiento de cada vez que ocurra algún tipo de error en los discos.

8. También se tiene que modificar una línea en el archivo:

#cd /etc/mail/sendmail.mc

Se cambia

```
DAEMON_OPTIONS(`Port=smtp,Addr=127.0.0.1, Name=MTA')
```

por:

```
DAEMON_OPTIONS(`Port=smtp, Name=MTA')dnl
```

9. Por último se compila, para que las modificaciones realizadas se tomen en cuenta:

```
make -C /etc/mail
```

Monitoreo con Logs

Además de las herramientas de monitoreo mencionadas arriba, se puede utilizar otro recurso que también es de gran ayuda para saber la procedencia de un error dentro de las bases de datos.

Un *log* es un registro de actividad de un sistema, que se guarda en un fichero de texto, al que se le van añadiendo líneas a medida que se realizan acciones sobre el sistema.

PostgreSQL cuenta con soporte para *logs,* en el archivo postgres.conf en la sección de re portes de errores y acceso, se encuentran distintos parámetros configurables, tales parámetros proporcionarán que se tenga un registro de los *logs* de los nodos del *cluster*, para el determinado momento que se presente una falla en alguno de los nodos se pueda conocer en qué momento ocurrió la falla y la causa de la misma.

A continuación se muestra la sección del archivo postgres.conf donde se configura el manejo de los *logs*:

```
#------------------------------------------------------------------------------
# ERROR REPORTING AND LOGGING
#------------------------------------------------------------------------------

# - Where to Log -

log_destination = 'stderr'       # Valid values are combinations of
                                 # stderr, syslog and eventlog,
                                 # depending on platform.

# This is used when logging to stderr:
redirect_stderr = on                   # Enable capturing of stderr into log
                                 # files
                                 # (change requires restart)

# These are only used if redirect_stderr is on:

log_directory = 'pg_log'         # Directory where log files are written
                                 # Can be absolute or relative to PGDATA
log_filename = 'postgresql-%Y-%m-%d_%H%M%S.log' # Log file name pattern.
                                 # Can include strftime() escapes
log_truncate_on_rotation = on          # If on, any existing log file of the same
                                 # name as the new log file will be
                                 # truncated rather than appended to. But
                                 # such truncation only occurs on
                                 # time-driven rotation, not on restarts
                                 # or size-driven rotation. Default is
                                 # off, meaning append to existing files
                                 # in all cases.
log_rotation_age = 1d                  # Automatic rotation of logfiles will
                                 # happen after that time.  0 to
                                 # disable.
log_rotation_size = 10MB         # Automatic rotation of logfiles will
                                 # happen after that much log
                                 # output.  0 to disable.
```

Donde se indica la ubicación de los *logs,* el formato de la fecha como será nombrado el *log*, la activación de rotación del *logs* con duración de un día y capacidad de 10Mb para el mismo.

Con las herramientas mencionadas para monitorear distintos componentes del *cluster*, se logra prevenir fallas y tomar las precauciones debidas, antes de presentarse una situación crítica que ponga en riesgo la información contenida en el *cluster.*

Se tiene a Mon con el que se consigue monitorear el estado de los servicios que brinda el *cluster,* se cuenta con Power Alert herramienta que se encarga de apagar los equipos de manera correcta cuando ocurra una interrupción alargada de la energía eléctrica para evitar daños en los datos y en los dispositivos físicos, además se tiene Smart Mon Tools que se encarga de monitorear el estado de los discos físicos de los equipos y alertar cuando ocurra una falla en ellos, y por último se cuenta con el almacenamiento de *logs* que permite conocer dónde y cómo se efectuó un error en las bases de datos.

Conclusiones

La realización de este proyecto permitió implementar un nuevo tipo de tecnología, la tecnología de *cluster*, dicha tecnología ofrece un sistema de alta disponibilidad para las bases de datos que interactúan con la geDGAPA; misma de la que se carecía y producía tiempos fuera de servicio de las aplicaciones que forzosamente necesitaban establecer conexión con las bases de datos. Ahora el *cluster* satisface ampliamente las necesidades que se presentan en el manejo diario de los servicios que brinda la geDGAPA al personal académico de la Universidad.

Todos los componentes de *software* del *cluster* son de Licencia Pública General, lo que significa que es *software* libre de código abierto, por lo tanto, la implementación del *cluste*r tiene un mérito sobresaliente, porque hoy en día pocas son las instituciones que toman el riesgo de ir de la mano con el *software* libre y probar que funciona adecuadamente, ahorrando recursos en la compra de licencias de *software* comercial y soporte técnico; por ello , la DGAPA es una institución que va más allá, apoyando e impulsando nuevas tecnologías que fomenten el progreso de sí y de los servicios que brinda.

Para implementar el *cluster* era necesario contar con recursos de *hardware* y *software* más eficientes de los que se tenían; por dicho motivo fue necesario migrar el servidor de bases de datos a un servidor más potente y con mayor capacidad, un Sun Fire X4200; con este servidor se obtiene un mejor rendimiento, porque tiene más memoria RAM y un procesador de doble núcleo que ofrece más velocidad de respuesta a los procesos.

Para el *software*, se actualizó la versión del DBMS de PostgreSQL de la versión 7.4.6 a la 8.2.4, optimizando el funcionamiento del servidor de bases de datos, logrando que trabaje bien con grandes cantidades de datos y con muchos usuarios accediendo a la vez al sistema.

El *cluster* proporciona redundancia tanto en *hardware* como en *software*; en *hardware* porque se tienen tres servidores que se comunican y funcionan como uno solo de manera transparente para los usuarios. En *software,* los mismos servidores pueden actuar como replicador cuando sea necesario, es decir, cuando el nodo maestro no funcione adecuadamente y se realice un proceso de *failover,* cualquiera de los otros dos nodos es capaz de reemplazar al nodo maestro, porque cuenta con las herramientas y configuraciones necesarias para poder satisfacer las peticiones de consulta a la base de datos por parte de los clientes.

La planeación que se le dio a la realización de este proyecto permitió lograr satisfactoriamente la puesta en marcha del *cluster.* Cada una de las etapas del proyecto*,* fueron fundamentales para que todo el trabajo en conjunto resultará en el éxito de los objetivos planteados para resolver la problemática inicial, acerca de la sobrecarga del servidor de bases de datos, que mermaba el funcionamiento de las aplicaciones, al tener

que responder a todas las peticiones de acceso a las bases de datos en un tiempo inadecuado.

En resumen, las ventajas obtenidas de la realización de este proyecto son:

- Se tiene un sistema de replicación de bases de datos que proporciona alta disponibilidad de las mismas.
- Se tienen medidas preventivas, como el proceso de *failover,* que actúa ante la presencia de una falla en el nodo maestro.
- Se cuenta con una nueva versión de DBMS de PostgreSQL que permite mantener a la geDGAPA con un *software* actual.
- Con *software* de Licencia Pública General se brinda a la geDGAPA de una alta tecnología.
- Se implementaron estrategias nuevas de monitoreo que facilitan la prevención e identificación de fallas.
- El rendimiento de los sistemas que acceden a la base de datos es más ágil; porque ahora se tiene tres nodos que pueden responder a peticiones de lectura de los datos.

Con base en las pruebas realizadas, se concluye que el *cluster* es capaz de brindar la replicación de la base de datos dbgdgapa de manera exitosa, proporcionando redundancia de la misma en dos nodos extras disponibles para su uso cuando se necesite.

Los resultados fueron satisfactorios a medida de las necesidades a resolver de la dependencia, sin embargo, también hay que tener presente los componentes del *cluster* presentan debilidades y cuales otros pueden mejorarse.

El *cluster* depende de *software´s* que son independientes, pero que interactúan unos con otros, para lograr el resultado de ofrecer alta disponibilidad de la base de datos; pero al mismo tiempo se trabaja con *software* libre y qué implica ésto: que a diferencia del *software* comercial que se vende con licencia y soporte, el *software* libre no cuenta con soporte cuando se presente un problema en el funcionamiento del mismo; lo cual dificulta saber la causa y solución de un problema. Cabe señalar, que para este tipo de casos el mejor lugar para hallar la solución se encuentra en la *web*, en consecuencia, implica llevarse más tiempo para la solución de un problema, que una simple llamada a la compañía que nos proporciona el *software* como en el caso comercial.

Además, comparando de nuevo con el *software* comercial, cuando nos brinda una herramienta, también ofrece toda una interfaz para la configuración y administración de la misma. El *cluster* de la geDGAPA no tiene una sola consola de configuración y administración, sino que cuenta con *software´s* independientes, con consolas independientes que al propósito de cada una, brindan la capacidad de configurar y administrar los nodos del *cluster,* sería deseable poseer una sola consola para administrar

todo el *cluster*, esto no implica que con lo que se cuenta no funcione, simplemente que sería más práctico y sencillo el manejo del *cluster*; porque como se encuentra actualmente es necesario que el personal que se dedique a esta tarea tenga un conocimiento detallado de cómo funciona y se administra el *cluster* de alta disponibilidad.

Por último, la geDGAPA no tiene un *site* de cómputo, esto quiere decir que no se tiene un lugar totalmente equipado para los servidores, donde se cuente con las medidas necesarias para que al menos la ocurrencia de un error de *hardware* sea nula. Es decir, un lugar que cuente con la ventilación adecuada para que no se sobrecalienten los equipos, piso y techo falso para evitar que la humedad afecte los componentes de los equipos o una planta de luz que asegure la energía eléctrica a los servidores, que en consecuencia beneficiaría no solamente a los equipos del *cluster,* sino a todos los demás ocupados para el funcionamiento de la geDGAPA.

La finalidad de mencionar las ventajas y optimizaciones que se pueden realizar a través del desarrollo de este trabajo de tesis, solo desea informar al lector del panorama con el que se tuvo que trabajar y que con los recursos otorgados por la dependencia y el excelente grupo de trabajo del área de sistemas de la DGAPA, se logró un proyecto innovador de calidad, justo al nivel que la Universidad brinda y merece.

Bibliografía

- Emmanuel Cecchet,Anastassia Ailamki, George Candea, 2007, *Middleware*-based Database Replication: The Gaps between Theory and Practice.
- DataComparer 2007 for PostgreSQL User's manual.
- DBComparer 2007 for PostgreSQL User's manual.
- Gustavo Gómez Macías, 2008, Diseño de un cluster Beowulf para realizar cálculos Monte Carlo.
- Vincent Gozalbes Sanchis, 2007, Configuración de un firewall utilizando iptables.
- Antonio Hernández Rodríguez, 2003, Implementación de un sistema de alta disponibilidad basado en tecnología de cluster.
- Jan Wieck,2005, Slony-I System Overview.
- Jan Wieck, Slony-I a replication system for PostgreSQL.
- Michal Valenta,2008, Mechanism of replication whit Slony-I.
- Mikko Partio, 2007, Evaluation of PostgreSQL replication and load balancing implementations.
- Vallath Murali, 2004, Oracle real application clusters.

Mesografía

- http://clusterfie.epn.edu.ec/clusters/Definiciones/definiciones.html
  Componentes de un cluster.
- http://dgapa.unam.mx
- http://dgapa.unam.mx/org_nuevo09/orgral08.html
- http://dgapa.unam.mx/programas/a_papime/papime.html
- http://dgapa.unam.mx/programas/a_infocab/infocab.html
- http://dgapa.unam.mx/programas/a_pasd/pasd.html
- http://dgapa.unam.mx/programas/d_papiit/papiit.html
- http://dgapa.unam.mx/programas/e_pepasig/pepasig.html
- http://dgapa.unam.mx/programas/e_pride/pride.html
- http://dgapa.unam.mx/programas/f_paspa/paspa.html
- http://dgapa.unam.mx/programas/r_pun/pun.html
- http://dgapa.unam.mx/programas/r_rdunja/rdunja.html
- http://mon.wiki.kernel.org
  Mon – Service Monitoring Daemon.
- http://pgpool.projects.postgresql.org/pgpool-II/doc/pgpool-en.html
  Pgppool user´s manual.
- http://phppgadmin.sourceforge.net/
  phpPgAdmin.
- http://smartmontools.sourceforge.net/
  Smart Mon Tools Home Page
- http://usuarios.lycos.es/janjo/janjo1.html
  Protocolo TCP/IP por Miguel Alejandro Soto.

- http://www.apesol.org.pe
  Asociación Peruana de *Software* Libre.
- http://www.grulic.org.ar/linux.html
  Linux Information Sheet 1997 por Michael K. Johnson
- http://www.htmlpoint.com/sql/sql_04.htm
  Breve historia de SQL.
- http://www.maestrosdelweb.com/principiantes/cuidar/
  ¿Cómo alargar la vida de nuestro equipo de computación? por Christian Van Der Hents.
- http://www.netpecos.org/docs/mysql_postgres/x15.html,
  PostGreSQL vs MySQL.
- http://www.pello.info/filez/firewall/iptables.html#1
  IPTABLES Manual práctico por Pello Xabier Altardill Izura.
- http://www.postgresql.org.pe/articles/comparativa_dbms_libres.pdf
  Comparativa de PostgreSQL vs otros DBMS Libres por Ernesto Quiñones
- http://www.slideshare.net/chel0nline/tcpip-presentation
  TCP/IP vs Modelo OSI.
- http://www.slony.info/documentation/requirements.html
  System Requirements of Slony.
- http://www.slony.info/documentation/slonyintro.html#INTRODUCTION
  Slony-I 2.0.3_RC2 Documentation.
- http://www.tripplite.com/en/support/pdf/PowerAlertUserManual045.pdf
  User's Guide Power Alert *Software* Version 12.04.0045
- http://www.tripplite.com/en/support/PowerAlert/index.cfm
  Tripp-Lite Support.
- http://www.webexperto.com/articulos/articulo.php?cod=136
  Instalación de un servidor de PostgreSQL bajo Linux por Luis Tomás Wayar.

Apéndices

A.1 Glosario

A

Alta disponibilidad. Es la capacidad de proveer disponibilidad y confiabilidad en el momento que se necesite, esta puede ser provista mediante la detección de la falla de un nodo o servicio y reconfigurando el sistema apropiadamente para que la carga de trabajo pueda ser manejada por los restantes nodos del *cluster*.

Alto rendimiento. Es la capacidad que se tiene para ejecutar tareas que requieren de gran capacidad de cómputo.

ANSI. (*American National Standards Institute*) Instituto Nacional Americano de Estándares es una organización encargada de estandarizar ciertas tecnologías en EEUU. ANSI es una organización privada sin fines de lucro, que permite la estandarización de productos, servicios, procesos, sistemas y personal en Estados Unidos.

Arquitectura cliente/servidor. Es la tecnología que le da el usuario final la capacidad de accesar transparentemente a cualquier aplicación, datos, servicios de cómputo o algún otro recurso dentro de un grupo de trabajo a través de una organización en diversas plataformas.

B

Backup. Una copia de seguridad o *backup* en informática es un archivo digital, un conjunto de archivos o la totalidad de los datos considerados lo suficientemente importantes para ser conservados.

Backend. *Front-end y back-end* son términos que se relacionan con el principio y el final de un proceso. El *front-end* es la parte del *software* que interactúa con los usuarios y el *back-end* es la parte que procesa la entrada desde el *front-end.* La idea general es que el *front-end* sea el responsable de recolectar los datos de entrada del usuario, que pueden ser de muchas y variadas formas, y procesarlas de una manera conforme a la especificación que el *back-end* pueda usar. La conexión del *front-end y el back-end* es un tipo de interfaz.

Balanceo de cargas. Es la capacidad de repartir la carga de trabajo entre los nodos del *cluster*.

BD. Base de datos, es una serie de datos organizados y relacionados entre sí, los cuales son recolectados y explotados por un sistema de gestión o de administración de base de datos (DBMS).

Beowulf. En 1994, se integró el primer *cluster* de PCs en el Centro de Vuelos Espaciales Goddard de la NASA, para resolver problemas computacionales que aparecen en las ciencias de la Tierra y el Espacio. Los pioneros de este proyecto fueron Thomas Sterling, Donald Becker y otros científicos de la NASA. El *cluster* de PCs desarrollado tuvo una eficiencia de 70 megaflops. Los investigadores de la NASA le dieron el nombre de Beowulf a este *cluster*, en honor del héroe de las leyendas medievales, quien derrotó al monstruo gigante Grendel. Existen tantas definiciones de Beowulf, algunos autores afirman que se debe llamar Beowulf sólo aquellos computadores construidos de la misma forma que el computador original de la NASA. Sin embargo, hay autores que coinciden con la siguiente definición: “Beowulf es una arquitectura conformada por múltiples computadores que puede usarse para computación paralela con infraestructura de *software* libre (Linux)”.

BISON. Es un programa generador de analizadores sintácticos convierte la descripción formal de un lenguaje, escrita como una gramática libre de contexto, en un programa en C, C++, o Java que realiza análisis sintáctico.

Bootear. En informática, la secuencia de arranque, (*boot* o *booting* en inglés) es el proceso que inicia el sistema operativo cuando el usuario enciende una computadora. Se encarga de la inicialización del sistema y de los dispositivos.

BSD. Es la licencia de *software* otorgada principalmente para los sistemas BSD (*Berkeley Software Distribution*). Pertenece al grupo de licencias de *software* Libre. Esta licencia tiene menos restricciones en comparación con otras como la GPL estando muy cercana al dominio público. La licencia BSD al contrario que la GPL permite el uso del código fuente en *software* no libre.

Bytes. Un byte es la unidad fundamental de datos en los ordenadores personales, un byte son ocho bits contiguos. El byte es también la unidad de medida básica para memoria, almacenando el equivalente a un carácter.

C

Cache. Una memoria caché es una memoria en la que se almacena una serie de datos para su rápido acceso.

Cascada. Característica de Slony-I de permitir a un esclavo ser a su vez maestro para otro servidor.

Cluster. Conjunto de pc's o estaciones de trabajo (llamados nodos) que interconectadas con dispositivos de alta velocidad por alguna tecnología de red, trabajan como un solo recurso integrado de cómputo y aparecen ante clientes y aplicaciones como un solo sistema, con el objetivo de satisfacer alguna necesidad de: alta disponibilidad, balanceo de carga, alto rendimiento o alguna combinación de las anteriores.

Clustering. Termino común para identificar el mecanismo de distribuir un servicio sobre un número de servidores para incrementar la tolerancia a fallas y soportar mayores cargas que las que podría soportar un servidor simple. Es usado para aplicaciones de gran escala y de misión crítica donde no puede haber tiempos muertos.

Commit. Se refiere a la idea de hacer que un conjunto de cambios "tentativos, o no permanentes" se conviertan en permanentes. Un uso popular es al final de una transacción de base de datos. Una sentencia COMMIT en SQL finaliza una transacción de base de datos dentro de un sistema gestor de base de datos relacional (RDBMS) y pone visibles todos los cambios a otros usuarios.

Consulta. Cualquier sentencia SQL que manipule datos.

Constraints. Las *constraints* son las encargadas de asegurar la integridad referencial en la base de dato. Tipos de *contraints*: CHECHK, NOT NULLM UNIQUE KEY, PRIMARY KEY y FOREIGN KEY.

CPU. (*Central Processing Unit*)Unidad de Proceso Central, se pronuncia como letras separadas. La CPU es el cerebro del ordenador. A veces es referido simplemente como el procesador o procesador central, la CPU es donde se producen la mayoría de los cálculos. En términos de potencia del ordenador, la CPU es el elemento más importante de un sistema informático.

D

DBMS.(*Data Base Management System*)Son las siglas en inglés para los Sistemas de Gestión de Bases de Datos y es un *software* que controla la organización, el almacén, la recuperación, la seguridad e integridad de datos en una base de datos.

DDL (*Data Definition Language*) permite crear y definir nuevas bases de datos, campos e índices.

DML (*Data Manipulation Language*) permite generar consultas para ordenar, filtrar y extraer datos de la base de datos.

Downtime. El término *downtime* es usado para referirse a periodos de tiempo cuando un sistema no se encuentra disponible o fuera de servicio.

DRAM.(*Dynamic Random Access Memory*)En su traducción Memoria Dinámica de Acceso Aleatorio es una memoria fabricada a partir de transistores acoplados con condensadores siendo éstos los que dependiendo de su estado (cargado o descargado) representan los unos y ceros a almacenar.

Dump. Es un comando de PostgreSQL usado regularmente para hacer un procedimiento de respaldo desde un *host* remoto que puede accesar a la base de datos.

E

ECC. (*Error checking and correction*) Chequeo y corrección de errores se basa en un algoritmo más complejo y se utiliza en computadoras de gama alta, como servidores de red. El sistema trabaja en conjunción con el controlador de memoria, y anexa a los bits de datos, los bits ECC, que son almacenados junto con los de datos. Estos bits extras, junto con la decodificación correspondiente, sirven para realizar la comprobación en el momento de la lectura.

Escalabilidad. La escalabilidad es típicamente definida como la habilidad de un sistema de adaptarse a las necesidades de crecimiento de un negocio que puedan surgir en un periodo de tiempo.

Esclavo. Servidor que recibe modificaciones del maestro.

Esquema. Es la descripción lógica de la base de datos, proporciona los nombres de las entidades y sus atributos especificando las relaciones que existen entre ellos.

Estabilidad. Es la capacidad de un equipo de hacer frente a volúmenes de trabajo cada vez mayores, sin dejar por ello de prestar un nivel de rendimiento aceptable.

F

Failback. Ocurre cuando sucede una falla en la replica más reciente y entonces los usuarios son reubicados a la réplica anterior a que ocurriera la falla.

Failover. Es la capacidad para los usuarios de una base de datos ser cambiados a otro nodo de bases de datos que contenga una réplica de los datos sin importar en que nodo hubiera ocurrido la falla y todo esto de manera transparente para el usuario de la BD.

***Fast Ethernet*.**También llamado *Ethernet* de alta velocidad es el nombre de una serie de estándares de IEEE de redes *Ethernet* de 100 Mbps.

Firewall. Dispositivo que filtra el tráfico entre redes.

FLEX. Es un programa generador de analizadores sintácticos acompañado normalmente de Bison.

Forward. Cadena de manipulación de paquetes, usado en los paquetes no generados localmente.

Frontend. *Front-end* y *back-end* son términos que se relacionan con el principio y el final de un proceso. El front-end es la parte del *software* que interactúa con el o los usuarios y el *back-end* es la parte que procesa la entrada desde el front-end. La idea general es que el *front-end* sea el responsable de recolectar los datos de entrada del usuario, que pueden ser de muchas y variadas formas, y procesarlas de una manera conforme a la especificación que el *back-end* pueda usar. La conexión del *front-end* y el *back-end* es un tipo de interfaz.

FTP. (*File Transfer Protocol*) en su traducción Protocolo de Transferencia de Archivos es un protocolo de red para la transferencia de archivos entre sistemas conectados a una red TCP, basado en la arquitectura

cliente-servidor. Desde un equipo cliente se puede conectar a un servidor para descargar archivos desde él o para enviarle archivos, independientemente del sistema operativo utilizado en cada equipo.

G

Gb. Un gigabyte es una unidad de medida informática cuyo símbolo es el GB, y puede equivalerse a 2^{30} bytes.

Gigabit ethernet. También conocida como GigaE, es una ampliación del estándar Ethernet que consigue una capacidad de transmisión de 1 gigabit por segundo, correspondientes a unos 1000 megabits por segundo de rendimiento contra unos 100 de *Fast Ethernet*.

GNU. El proyecto GNU fue iniciado por Richard Stallman con el objetivo de crear un sistema operativo completamente libre: el sistema GNU.

GPL. (*General Public License*) Licencia Pública General fue creada por la Fundación de *Software* Libre *(Free Software Foundation)* y orientada principalmente a los términos de distribución, modificación y uso de *software* libre.

H

***Hardware*.** Corresponde a todas las partes físicas y tangibles de una computadora: sus componentes eléctricos, electrónicos, electromecánicos y mecánicos; sus cables, gabinetes o cajas, periféricos de todo tipo y cualquier otro elemento físico involucrado.

Hearbeat. Es un demonio que proporciona al *cluster*, la comunicación entre los clientes y los servicios que brinda, además de permitir a los clientes saber sobre la presencia o la ausencia de un proceso en algún nodo o computadora remota mediante la capacidad de intercambiar mensajes entre ellos.

Host. Se le llama *host* a aquel dispositivo conectado a una red que ofrece servicios a otros ordenadores conectados a dicha red. Puede ser un ordenador, un servidor de archivos, un dispositivo de almacenamiento por red, una máquina de fax, impresora, etc.

Hostname. Al nombre único de un equipo conectado a una red y conocido en inglés como *hostname*. Este nombre, ayuda al administrador de la red a identificar las máquinas sin tener que memorizar una dirección IP para cada una de ellas.

Http. *(HyperText Transfer Protocol)* es el protocolo de transferencia de hipertexto y es el protocolo usado en cada transacción de la Web. HTTP define la sintaxis y la semántica que utilizan los elementos *software* de la arquitectura *web* (clientes, servidores, *proxies*) para comunicarse. Es un protocolo orientado a transacciones y sigue el esquema petición-respuesta entre un cliente y un servidor.

HSC. Las unidades especiales de control, llamados como **HSCs** en su traducción Almacenamiento Jerárquico de Controladores *(Hierarchical Storage Controllers).* EL HSC maneja el disco y los dispositivos de entrada-salida de una pc.

Hub. También llamado concentrador es un equipo de redes que permite conectar entre sí otros equipos y retransmite los paquetes que recibe desde cualquiera de ellos a todos los demás.

I

IDE.(*Integrated Development Enviroment*) es un conjunto de programas que corren desde una sencilla interfaz de usuario.

Informix. El DBMS Informix fue concebido y diseñado por Roger Sippl a finales de los años 1970. La compañía Informix fue fundada en 1980, salió a bolsa en 1986 y durante parte de los años 1990 fue el segundo sistema de bases de datos más popular después de Oracle. Sin embargo, su éxito no duró mucho y para el año 2000 una serie de tropiezos en su gestión había debilitado seriamente a la compañía desde el punto de vista financiero.

Input. Cadena de manipulación de paquetes que afecta a todos los paquetes dirigidos a un *host.*

Integridad. Proceso de prevenir la eliminacióno adulteración en una base de datos.

Intel. Es una empresa multinacional que fabrica microprocesadores, circuitos integrados especializados tales como circuitos integrados auxiliares para placas base de computadora y otros dispositivos electrónicos.

Internet. Conjunto de redes de computadoras creada a partir de redes de menor tamaño, cuyo origen reside en la cooperación de dos universidades estadounidenses. Es la red global compuesta de miles de redes de área local(LAN) y de redes de área extensa (WAN) que utilizan TCP/IP para proporcionar comunicaciones de ámbito mundial.

IOS. (*Internetwork Operating System*) Sistema Operativo de Interconexión de Redes es un sistema operativo creado por Cisco Systems para programar y mantener equipos de interconexión de redes informáticas como *switches* (conmutadores) y *routers* (enrutadores).

ISO. (*International Organization for Standardization*) la Organización Internacional para la Estandarización o ISO, nacida tras la Segunda Guerra Mundial (23 de febrero de 1947), es el organismo encargado de promover el desarrollo de normas internacionales de fabricación, comercio y comunicación para todas las ramas industriales a excepción de la eléctrica y la electrónica. Su función principal es la de buscar la estandarización de normas de productos y seguridad para las empresas u organizaciones a nivel internacional.

Iptables. Sistema de *firewall* vinculado al kernel de linux.

J

Java. Es una plataforma virtual de *software* desarrollada por Sun Microsystems, de tal manera que los programas creados en ella puedan ejecutarse sin cambios en diferentes tipos de arquitecturas y dispositivos computacionales

K

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

L

Linux. Es un núcleo de sistema operativo libre tipo Unix. Es utilizado por la familia de sistemas operativos GNU/Linux. Lanzado bajo la licencia pública general de GNU y desarrollado gracias a contribuciones provenientes de todo el mundo, Linux es uno de los ejemplos más notables de *software* libre.

Logs. Un log es un registro de actividad de un sistema, que generalmente se guarda en un fichero de texto, al que se le van añadiendo líneas a medida que se realizan acciones sobre el sistema.

M

MANGLE. La regla mangle permite la reescritura completa de paquetes (o tramas completas), combinando la funcionalidad dentro de las cadenas dentro de filter (INPUT, OUTPUT y FORWARD) y nat (PREROUTING y POSTROUTING).

Maestro. Servidor que acepta modificaciones del usuario.

Mainframe. Una computadora central o *mainframe* es una computadora grande, potente y costosa usada principalmente por una gran compañía para el procesamiento de una gran cantidad de datos; por ejemplo, para el procesamiento de transacciones bancarias.

Mb. El Megabit es una unidad de medida de información muy utilizada en las transmisiones de dato, el Megabit que equivale a 10^6 (1.000.000) bits.

Megabyte. Un Megabyte son 1.048.576 (2 a la vigésima potencia) bytes.

Mflops. Megaflops son millones de operaciones de punto flotante por segundo.

Microprocesador. El microprocesador es el componente encargado de interpretar datos y ejecutar las instrucciones de los programas, también coordina a los demás componentes de una computadora. El microprocesador es comúnmente llamado el cerebro de la computadora pero dada la incapacidad de este para pens*a*r una descripción más acertada sería decir que el microprocesador es una calculadora extremadamente veloz (miles de millones de operaciones por segundo) con la capacidad de realizar operaciones aritméticas simples y almacenar sus resultados.

Middleware. Es el *software* intermediario que se sitúa entre una aplicación y un sistema operativo y permite a los procesos que se ejecutan en una o más computadoras interactuar entre sí a través de una red de datos.

Minix. Es un clon del sistema operativo Unix distribuido junto con su código fuente y desarrollado por el profesor Andrew S. Tanenbaum en 1987. La última versión oficial de Minix es la 3.1.2, publicada el 8 de Mayo de 2006.

MIT. *(Massachusetts Institute of Technology)* el Instituto Tecnológico de Massachusetts es una de las principales instituciones dedicadas a la docencia y a la investigación en Estados Unidos, especialmente en ciencia, ingeniería y economía. El Instituto está situado en Cambridge, Massachusetts, y cuenta con numerosos premios Nobel entre sus profesores y antiguos alumnos. MIT es considerada como una de las mejores universidades de ciencia e ingeniería del mundo.

Mon. Es una herramienta para supervisar la disponibilidad de servicios, y enviar alarmas de eventos definidos previamente en la configuración.

MTTF*. (Mean Time To Failure)* es el tiempo promedio de una falla, que se utiliza para determinar el porcentaje de disponibilidad de un sistema.

MTTR. *(Mean Time To Repair)* es el tiempo promedio para reparar una falla, que se utiliza para determinar el porcentaje de disponibilidad de un sistema.

Multiplataforma. Es un término usado para referirse a los programas, sistemas operativos, lenguajes de programación, u otra clase de *software*, que puedan funcionar en diversas plataformas, como Windows, Linux entre otros.

Multiprocesador. Se denomina multiprocesador a un ordenador que cuenta con dos o más microprocesadores (CPUs).

Multitarea. Es una característica de un sistema operativo moderno. Permite que varios procesos sean ejecutados al mismo tiempo compartiendo uno o más procesadores.

N

NAT. *(Network Address Translation)* Traducción de Dirección de Red es un mecanismo utilizado por *routers* IP para intercambiar paquetes entre dos redes que se asignan mutuamente direcciones incompatibles. Consiste en convertir en tiempo real las direcciones utilizadas en los paquetes transportados. También es necesario editar los paquetes para permitir la operación de protocolos que incluyen información de direcciones dentro de la conversación del protocolo.

Namespace. Es un término informático que se utiliza en programación, se refiere a una colección de nombres de entidad definidos por el programador, se puede resumir un *namespace* cómo un conjunto de nombres en el cual todos los nombres son únicos.

Nodo. Pueden ser simples computadoras, sistemas multi-procesador o estaciones de trabajo.

NTP. (*Network Time Protocol)* es un protocolo de Internet ampliamente utilizado para transferir el tiempo a través de una red. NTP es normalmente utilizado para sincronizar el tiempo en clientes de red a una hora precisa.

O

Output. Cadena de manipulación que afecta a los paquetes generados localmente.

***Open source*.** Código abierto es el término con el que se conoce al *software* distribuido y desarrollado libremente. El código abierto tiene un punto de vista más orientado a los beneficios prácticos de compartir el código que a las cuestiones morales y/o filosóficas las cuales destacan en el llamado *software* libre.

P

Path. Una ruta en inglés *path* señala la localización exacta de un archivo o directorio mediante una cadena de caracteres concreta.

PERL. *(Practical Extraction and Report Language)* es un lenguaje de programación desarrollado por Larry Wall inspirado en otras herramientas de UNIX como son: sed, grep, awk, c-shell, para la administración de tareas propias de sistemas UNIX.

PGCompare. Es un *software* que permite verificar que el número de registros de la base origen sea el mismo en cantidad y contenido en la base destino.

***Pg_dump*.** Es el comando utilizado por PostgreSQL para respaldar una base de datos.

***Pg_restore*.** Es el comando utilizado por PostgreSQL para restaurar una base de datos proveniente de un *pg_dump.*

PHP. PHP es un lenguaje interpretado de propósito general ampliamente usado y que está diseñado especialmente para desarrollo *web* y puede ser incrustado dentro de código HTML. Generalmente se ejecuta en un servidor *web*, tomando el código en PHP como su entrada y creando páginas *web* como salida.

Ping. La utilidad *ping* comprueba el estado de la conexión con uno o varios equipos remotos por medio de los paquetes de solicitud de eco y de respuesta de eco (ambos definidos en el protocolo de red ICMP) para determinar si un sistema IP específico es accesible en una red.

***Polling*.** Es una actividad similar al *ping,* donde un mensaje es enviado a un dispositivo y donde la respuesta es examinada. Si una respuesta es exitosa es recibida entonces el *polling* es exitoso, de lo contrario se determina que ocurre un problema.

POP3. *(Post Office Protocol)* Es el protocolo utilizado en clientes locales de correo para obtener los mensajes de correo electrónico almacenados en un servidor remoto. La mayoría de los suscriptores de los proveedores de Internet acceden a sus correos a través de POP3.

PostGIS. Es un módulo que añade soporte de objetos geográficos a la base de datos objeto-relacional PostgreSQL, convirtiéndola en una base de datos espacial para su utilización en sistemas de información Geográfica.

PostgreSQL. Es un gestor de bases de datos basado en el modelo relacional, aunque incorpora algunos conceptos del modelo Orientado a Objetos, tales como la herencia. Usa un subconjunto ampliado de SQL como lenguaje de consulta y está implementado siguiendo la arquitectura cliente/servidor. PostgreSQL ofrece gran variedad de herramientas y librerías para acceder a las bases de datos.

Postrouting. Cadena de manipulación de paquetes que pude modificarlo antes de salir a la red.

Protocolo. Es un conjunto de reglas usadas por computadoras para comunicarse unas con otras a través de una red. Un protocolo es una convención o estándar que controla o permite la conexión, comunicación, y transferencia de datos entre dos puntos finales.

Protocolo IP. (*Internet Protocol*) es un protocolo no orientado a conexión usado tanto por el origen como por el destino para la comunicación de datos a través de una red de paquetes conmutados.

Prerouting. Cadena de manipulación de paquetes que los modifica tan pronto como lleguen al *firewall* y antes de enrutarlos.

Q

Queries. En base de datos, *query* significa consulta, entonces, un *query* es una búsqueda o pedido de datos almacenados en una base de datos.

R

RAID. *(Redundant Array of Independent Disks)* conjunto redundante de discos independientes, o hace referencia a un sistema de almacenamiento que usa múltiples discos duros entre los que distribuye o replica los datos.

RAM. *(Random Access Memory)* Memoria de Acceso Aleatorio que almacena instrucciones y datos de manera temporal para realizar tareas dentro de una computadora.

Replicación. Capacidad para distribuir sincronizadamente bases de datos por rutinas de copiado a una o más bases de datos en otros servidores en la red.

Replicación asíncrona. Considera que las operaciones de escritura son finalizadas cuando la operación se termina en el nodo maestro principal, y el nodo remoto es actualizado también pero con cierto tiempo de retraso.

Replicación síncrona. En la replicación síncrona el contenido de las bases de datos que se replican es siempre el mismo sin importar las circunstancias.

Restore. Es un comando de PostgreSQL usado regularmente después del comando dump para hacer un procedimiento de restauracion de una base de datos desde un *host* remoto que puede accesar a la base de datos.

Rol. Un rol de base de datos no es más que una agrupación de permisos de sistema y de objeto.

***Rolled back*.** Es la transacción que mueve hacia atrás una transacción explícita o implícita al principio de la misma o en un punto específico dentro de la transacción.

Root. En sistemas operativos del tipo Unix, *root* es el nombre convencional de la cuenta de usuario que posee todos los derechos, es también llamado superusuario, normalmente esta es la cuenta de administrador.

Rpms. Revoluciones por minuto.

Router. El enrutador (*router)*, direccionador, ruteador o encaminador es un dispositivo de *hardware* para interconexión de red de ordenadores que opera en la capa tres, permite asegurar el enrutamiento de paquetes entre redes o determinar la ruta que debe tomar el paquete de datos.

S

SATA. (*Serial Advanced Technology Attachment*) es una interfaz de transferencia de datos entre la placa base y algunos dispositivos de almacenamiento, como puede ser el disco duro, u otros dispositivos de altas prestaciones que están siendo todavía desarrollados. Serial ATA sustituye a la tradicional Parallel ATA o P-ATA. SATA proporciona mayores velocidades, mejor aprovechamiento cuando hay varios discos, mayor longitud del cable de transmisión de datos y capacidad para conectar discos en caliente (con la computadora encendida).

Script. Conjunto de instrucciones. Es muy utilizado para la administración de sistemas UNIX. Son ejecutados por un intérprete de línea de órdenes y usualmente son archivos de texto.

SCSI. *(Small Computers System Interface)* Sistema de Interfaz para Pequeñas Computadoras, es un interfaz estándar para la transferencia de datos entre distintos dispositivos del bus de la computadora.

***Select*.** La sentencia SELECT nos permite consultar los datos almacenados en una tabla de la base de datos.

Sequence. Una secuencia es un objeto que se utiliza para generar una secuencia de números. Esto puede ser útil cuando se necesita crear un número único que actué como llave primaria.

Set. Es un conjunto de datos que son replicados entre los nodos.

Servidor. Es el proceso que atiende a las peticiones de los procesos del cliente y manda las respuestas apropiadas.

SGBD. (*DataBase Management System)* Los sistemas de gestión de base de datos son un tipo de *software* muy específico, dedicado a servir de interfaz entre la base de datos, el usuario y las aplicaciones que la utilizan.

***Shareware*.** Modalidad de distribución de *software*, tanto como juegos como programas utilitarios, y aplicaciones ofimáticas para que el usuario pueda evaluar de forma gratuita el producto, por un tiempo especificado, aunque también las limitaciones pueden estar en algunas de las formas de uso o las capacidades finales.

Shell. Es un programa informático que actúa como interfaz de usuario para comunicar al usuario con el sistema operativo mediante pantalla completa o ventana que espera órdenes escritas por el usuario. Un *shell* de Unix es una *shell* de línea de comando que permite a los usuarios controlan el funcionamiento de la

computadora introduciendo la instrucción en forma de texto, a fin de que un intérprete de líneas de comandos lo ejecute.

Sistema. Es un conjunto de partes o elementos organizados y relacionados que interactúan entre sí para lograr un objetivo.

Sistema modular. Se refiere a un sistema desarrollado por módulos o secciones independientes que conforman un sistema en su totalidad.

Sistema Operativo. Es un conjunto de programas de computación destinados a realizar muchas tareas entre las que destaca la administración eficaz de sus recursos.

Slon. Demonio de Slony que se encarga de ejecutar la replicación en el nodo.

Slonik. Es el procesador de los *scripts* generados por los eventos de replicación cuando ocurre un cambio en la configuración de Slony.

Slony. *Software* de replicación maestro-esclavo que permite replicar grandes bases de datos con un numero mínimo de nodos esclavos.

SMTP. *(Simple Mail Transfer Protocol)* Protocolo Simple de Transferencia de correo, es un protocolo de la capa de aplicación. Protocolo de red basado en texto utilizado para el intercambio de mensajes de correo electrónico entre computadoras u otros dispositivos (PDA's, teléfonos móviles, etc.).

SNMP. El Protocolo Simple de Administración de Red o SNMP (*Simple Network Management Protocol*) es un protocolo de la capa de aplicación que facilita el intercambio de información de administración entre dispositivos de red. SNMP permite a los administradores supervisar el desempeño de la red, buscar y resolver sus problemas, y planear su crecimiento.

Solaris. Es un sistema operativo de tipo Unix desarrollado por Sun Microsystems.

SQL. (Structured Query Language)Lenguaje utilizado para interrrogar y procesar datos en una base de datos relacional. Los comandos SQL pueden ser utilizados para trabajar interactivamente con una base de datos o pueden ser incrustados en un lenguaje de programación para interactuar con una base de datos.

***Software*.** El *software* es un ingrediente indispensable para el funcionamiento de la computadora. Está formado por una serie de instrucciones y datos, que permiten aprovechar todos los recursos que la computadora tiene, de manera que pueda resolver gran cantidad de problemas.

SRAM. *(Static Random Access Memory)* Memoria de Acceso Aleatorio Estática es una memoria fabricada a partir de componentes que no requieren de refresco, logrando así que la memoria pueda trabajar casi a la misma velocidad del microprocesador.

SSH. *(Secure Shell)* Intérprete de órdenes seguro es el nombre de un protocolo y del programa que lo implementa, y sirve para acceder a máquinas remotas a través de una red.

SSI. (*Single System Image*) genera la sensación al usuario de que utiliza una única computadora muy potente.

SSL. El protocolo SSL es un sistema diseñado y propuesto por Netscape Communications Corporation. Se encuentra en la pila OSI entre los niveles de TCP/IP y de los protocolos HTTP, FTP, SMTP, etc. Proporciona sus servicios de seguridad cifrando los datos intercambiados entre el servidor y el cliente.

Stratum. Es el nivel en el que se encuentra la comunicación de un servidor con un nodo dentro del protocolo NTP.

Supercomputadora. También llamado Superordenador es una computadora con capacidades de cálculo muy superiores a las comúnmente disponibles de las máquinas de escritorio de la misma época en que fue construida. Hoy se utiliza en ingeniería, medicina y ciencia.

Superusuario. El administrador del sistema en los sistemas UNIX/Linux se denomina superusuario. El superusuario es el responsable de la administración y configuración de todo el sistema, y el único que tiene permisos para añadir nuevos usuarios, instalar aplicaciones, configurar dispositivos, etc.

Switch. Un conmutador o *switch* es un dispositivo digital de lógica de interconexión de redes de computadores que opera en la capa 2 (nivel de enlace de datos) del modelo OSI. Su función es interconectar dos o más segmentos de red, de manera similar a los puentes (*bridges*), pasando datos de un segmento a otro de acuerdo con la dirección MAC de destino de las tramas en la red.

Switch-over*.* Acción de tomar un nodo esclavo el papel de nodo maestro.

SYNC**.** Es un grupo de transacciones que son aplicadas a los nodos esclavos de las actualizaciones realizadas en el nodo maestro.

T

TCP. (*Transmission Control Protocol*) Protocolo de Control de Transmisión es uno de los protocolos fundamentales en Internet. El protocolo garantiza que los datos serán entregados en su destino sin errores y en el mismo orden en que se transmitieron. También proporciona un mecanismo para distinguir distintas aplicaciones dentro de una misma máquina, a través del concepto de puerto. TCP da soporte a muchas de las aplicaciones más populares de Internet, incluidas HTTP, SMTP, SSH y FTP.

Threads. Mediante el uso de varios *threads*, se consigue ejecutar varios procesos en paralelo, de forma que cuando uno de ellos esté esperando algún evento, permita que el microprocesador ejecute alguno de los otros *threads* en espera. Cuando el evento que el primer *thread* esperaba sucede, de nuevo se intercambian los *threads* para que el primer *thread* continúe su ejecución.

Triggers. Procedimiento en sql que se ejecuta cuando un registro es adicionado, modificado o borrado. Este es usado para mantener la integridad referencial en la base de datos.

Tunning. Afinamiento de una aplicación o sistema para dejarlo con el mejor funcionamiento.

U

UDP. (*User Datagram Protocol*) Es un protocolo del nivel de transporte basado en el intercambio de datagramas. Permite el envío de datagramas a través de la red sin que se haya establecido previamente una conexión, ya que el propio datagrama incorpora suficiente información de direccionamiento en su cabecera.

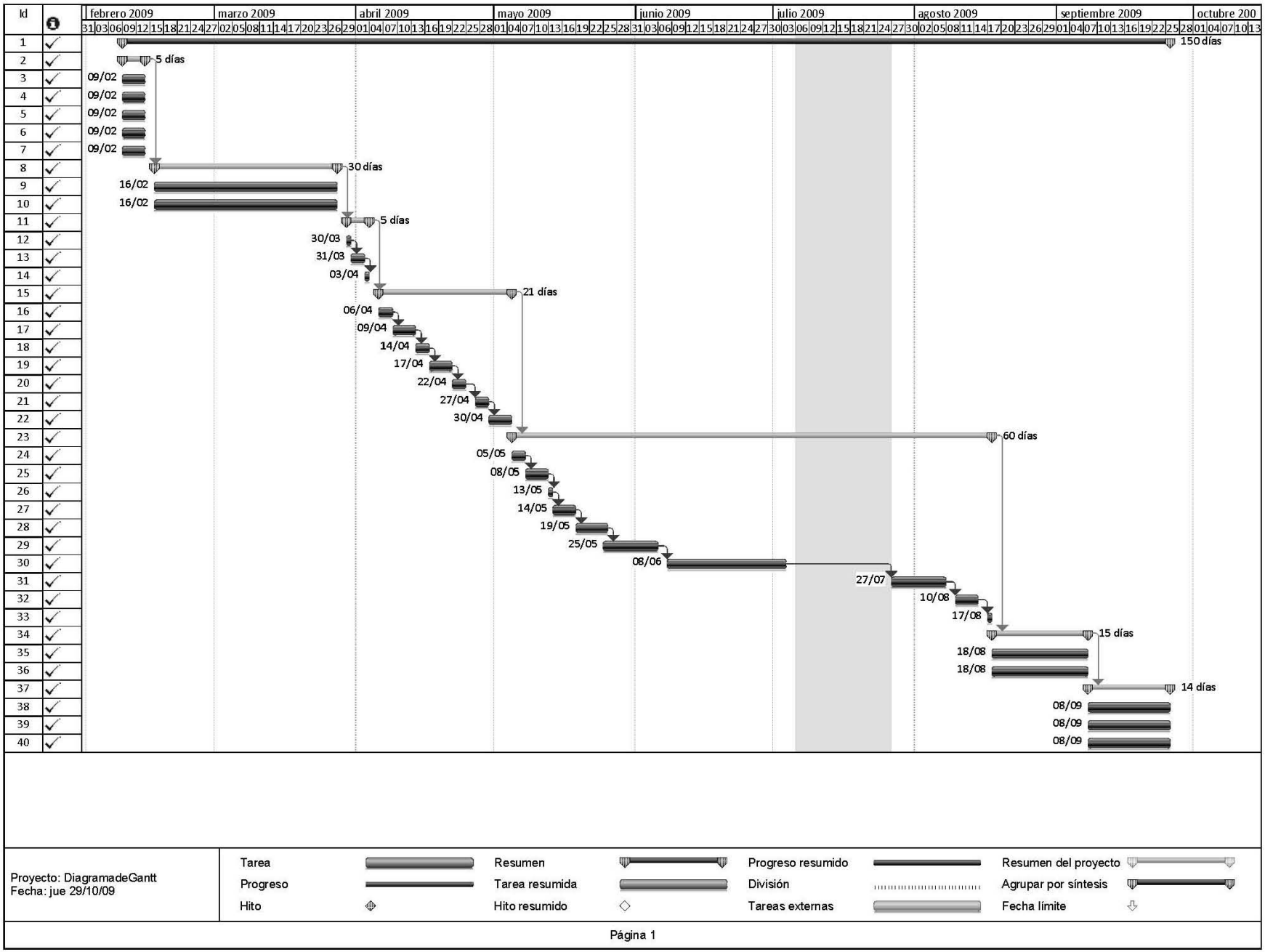
UNIX. Es un sistema operativo portable, multitarea y multiusuario; desarrollado, en principio, en 1969 por un grupo de empleados de los laboratorios Bell de AT&T. Es un sistema operativo de tiempo compartido, controla los recursos de una computadora y los asigna entre los usuarios. Permite a los usuarios correr sus programas. Controla los dispositivos de periféricos conectados a la máquina.

UPS. Sistema completo de respaldo de energía eléctrica capaz de mantener operando el equipo conectado a él, por un periodo de tiempo mientras no haya suministro eléctrico, con la finalidad de no perder la operación del equipo.

USB. (*Universal Serial Bus*) abreviado como USB, es un puerto que sirve para conectar periféricos a una computadora.

A.2 Diagrama de Gantt

		Nombre de tarea	Duración	Comienzo	Fin
1	✓	**Implementación de un cluster de alta disponibilidad de bases de dato**	150 días	lun 09/02/09	vie 25/09/09
2	✓	**ANÁLISIS**	5 días	lun 09/02/09	vie 13/02/09
3	✓	Problemas detectados.	5 días	lun 09/02/09	vie 13/02/09
4	✓	Propuesta de solución a la problemática.	5 días	lun 09/02/09	vie 13/02/09
5	✓	Resultados esperados.	5 días	lun 09/02/09	vie 13/02/09
6	✓	Recursos disponibles.	5 días	lun 09/02/09	vie 13/02/09
7	✓	Modelo de la solución.	5 días	lun 09/02/09	vie 13/02/09
8	✓	**INVESTIGACIÓN**	30 días	lun 16/02/09	vie 27/03/09
9	✓	Recopilación de información en libros.	30 días	lun 16/02/09	vie 27/03/09
10	✓	Recopilación de información en internet.	30 días	lun 16/02/09	vie 27/03/09
11	✓	**PLANIFICACIÓN DEL PROYECTO**	5 días	lun 30/03/09	vie 03/04/09
12	✓	Integración del equipo de trabajo.	1 día	lun 30/03/09	lun 30/03/09
13	✓	Asignación de tiempos para cada etapa del proyecto.	3 días	mar 31/03/09	jue 02/04/09
14	✓	Selección de los recursos de hardware y software.	1 día	vie 03/04/09	vie 03/04/09
15	✓	**DISEÑO**	21 días	lun 06/04/09	lun 04/05/09
16	✓	Diseño físico del cluster.	3 días	lun 06/04/09	mié 08/04/09
17	✓	Diseño de los sets de replicación de Slony	3 días	jue 09/04/09	lun 13/04/09
18	✓	Diseño de sincronización.	3 días	mar 14/04/09	jue 16/04/09
19	✓	Mecanismo de respaldo.	3 días	vie 17/04/09	mar 21/04/09
20	✓	Diseño del proceso de migración	3 días	mié 22/04/09	vie 24/04/09
21	✓	Diseño del monitoreo	3 días	lun 27/04/09	mié 29/04/09
22	✓	Diseño de implementación	3 días	jue 30/04/09	lun 04/05/09
23	✓	**IMPLEMENTACIÓN**	60 días	mar 05/05/09	lun 17/08/09
24	✓	Instalación y configuración del sistema operativo	3 días	mar 05/05/09	jue 07/05/09
25	✓	Configuración de la seguridad.	3 días	vie 08/05/09	mar 12/05/09
26	✓	Configuración de la sincronización del tiempo.	1 día	mié 13/05/09	mié 13/05/09
27	✓	Instalación y configuración de PostgreSQL.	3 días	jue 14/05/09	lun 18/05/09
28	✓	Proceso de respaldo	5 días	mar 19/05/09	lun 25/05/09
29	✓	Proceso de migración	10 días	lun 25/05/09	vie 05/06/09
30	✓	Instalación y configuración de Slony.	20 días	lun 08/06/09	vie 03/07/09
31	✓	Instalación y configuración de Pgpool.	10 días	lun 27/07/09	vie 07/08/09
32	✓	Instalación y configuración de herramientas de monitoreo.	5 días	lun 10/08/09	vie 14/08/09
33	✓	Puesta online.	1 día	lun 17/08/09	lun 17/08/09
34	✓	**PRUEBAS**	15 días	mar 18/08/09	lun 07/09/09
35	✓	Pruebas de replicación	15 días	mar 18/08/09	lun 07/09/09
36	✓	Pruebas de Failover	15 días	mar 18/08/09	lun 07/09/09
37	✓	**MONITOREO**	14 días	mar 08/09/09	vie 25/09/09
38	✓	Monitoreo con MON.	14 días	mar 08/09/09	vie 25/09/09
39	✓	Monitoreo con Power Alert.	14 días	mar 08/09/09	vie 25/09/09
40	✓	Monitoreo con SmartMonTools.	14 días	mar 08/09/09	vie 25/09/09


Id
febrero 2009
marzo 2009
abril 2009
mayo 2009
junio 2009
julio 2009
agosto 2009
septiembre 2009
octubre 200
150 días
5 días
09/02
09/02
09/02
09/02
09/02
30 días
16/02
16/02
5 días
30/03
31/03
03/04
21 días
06/04
09/04
14/04
17/04
22/04
27/04
30/04
60 días
05/05
08/05
13/05
14/05
19/05
25/05
08/06
27/07
10/08
17/08
15 días
18/08
18/08
14 días
08/09
08/09
08/09
Proyecto: DiagramadeGantt
Fecha: jue 29/10/09
Tarea
Progreso
Hito
Resumen
Tarea resumida
Hito resumido
Progreso resumido
División
Tareas externas
Resumen del proyecto
Agrupar por síntesis
Fecha límite
Página 1