IMPLEMENTACIÓN DE BASE DE DATOS HOMOGÉNEA DISTRIBUIDA EN EL AULA VIRTUAL MEIWEB

DANIEL DAVID DE LAS AGUAS CASTELLÓN

JUAN CARLOS ACEVEDO GUTIÉRREZ

UNIVERSIDAD INDUSTRIAL DE SANTANDER

FACULTAD DE INGENIERÍAS FISICO-MECÁNICAS

ESCUELA DE INGENIERÍA DE SISTEMAS E INFORMÁTICA

BUCARAMANGA

2015

IMPLEMENTACIÓN DE BASE DE DATOS HOMOGÉNEA DISTRIBUIDA EN EL AULA VIRTUAL MEIWEB

DANIEL DAVID DE LAS AGUAS CASTELLÓN

JUAN CARLOS ACEVEDO GUTIÉRREZ

Trabajo de grado para optar al título de

INGENIERO DE SISTEMAS E INFORMÁTICA

DIRECTOR

MSC. MANUEL GUILLERMO FLÓREZ BECERRA

UNIVERSIDAD INDUSTRIAL DE SANTANDER

FACULTAD DE INGENIERÍAS FISICO-MECÁNICAS

ESCUELA DE INGENIERÍA DE SISTEMAS E INFORMÁTICA

BUCARAMANGA

2015

AGRADECIMIENTOS

Es inevitable no llenarse la cabeza con una gran cantidad de rostros, de sonrisas, regaños y consejos que sin darse cuenta llegaron a hacer parte fundamental en la consecución de este proyecto, lo cual por obvias razones me es casi imposible de plasmar en tan poco espacio, a lo que se le puede sumar mi falta de cortesía. No es fácil nombrar a tantas personas en esta hoja, por lo cual si usted sabe que ha influido en mi vida, seguramente ese espaldarazo de apoyo fue recordado al momento de escribir. A pesar de lo anterior es imposible no agradecer de manera apremiante la labor de algunas personas que estuvieron desde el primer instante impulsando este anhelo con toda su fe:

"En primer lugar a DIOS, quien siempre se ha manifestado en mi vida y en cada reto impuesto, que no ha dejado a un lado, quien me carga y soporta todas mis dudas y desaciertos logrando hacerme llegar a la virtud en medio de la ignorancia de la que día a día intento salir.

A mis padres, esos dos seres soñadores, de entrega total y espectaculares desapegos, personas de humilde corazón y de bondadoso espíritu, que dieron sus palabras de aliento durante todo el trayecto de este proyecto y de la carrera misma, seres que sin lugar a dudas se merecen estos éxitos más que yo por su entrega y total amor.

A mis hermanos, quienes en los instantes más oscuros han logrado iluminar ese camino que a veces se volvía interminable.

A mi compañero de proyecto y al director del mismo, por su entrega, organización y herramientas que permitieron que hoy este culminado este documento.

A mi mejor amigo por su eterno e incondicional apoyo desde hace ya varios años, por su gran sonrisa y limpio actuar, que me dieron impulso en momentos interminables.

A cada uno de los integrantes de la Coral Universitaria UIS, que durante estos años han sacado lo mejor de mí, que me han brindado sensibilidad y sentido de pertenencia, disciplina y pasión, apoyo incondicional, especialmente a su director, el maestro Juan Manuel Hernández Morales, por todo el conocimiento que ha impartido con gran amistad y desinterés.

Finalmente a cada libro, a cada poesía, canción o cuento, que sin temor a equivocarme están inmersos en cada letra de este libro."

Juan Carlos Acevedo Gutiérrez

Agradecimientos

Agradezco a Dios por darme la entereza, sabiduría, y motivación suficiente para afrontar este desafío y culminarlo de la mejor manera.

A mis padres por brindarme su apoyo incondicional, por darme la oportunidad de ser alguien en la vida e inculcarme buenos valores para ser una persona de bien.

A mis abuelas por todo el amor que me han dado y al resto de mi familia por cada una de las experiencias compartidas.

A mi novia por su apoyo, amor, cariño y compañía en la consecución de este logro.

A mi compañero y director de proyecto por su valiosa colaboración, que permitió sacar adelante este proyecto.

A mis compañeros y profesores por compartir conmigo sus conocimientos.

Daniel David De las Aguas Castellón.

CONTENIDO

LISTA DE FIGURAS

pág.

LISTA DE TABLAS

pág.

LISTA DE IMÁGENES

pág.

LISTA DE ANEXOS

pág.

RESUMEN

TÍTULO: IMPLEMENTACIÓN DE BASE DE DATOS HOMOGÉNEA DISTRIBUIDA EN EL AULA VIRTUAL MEIWEB.*

AUTORES: DE LAS AGUAS CASTELLÓN, Daniel David**

ACEVEDO GUTIÉRREZ, Juan Carlos**

PALABRAS CLAVES: Base de datos distribuida, MySQL Cluster, Replicación.

DESCRIPCIÓN: MEIWEB es un aula virtual que presta servicios académicos a la Universidad Industrial de Santander (UIS) usando tecnología en la nube, lo cual implica el manejo de una gran cantidad de datos almacenados en una base de datos centralizada; con el objetivo de optimizar y aumentar la confiabilidad y la alta disponibilidad del aula virtual se hace necesario la existencia de varias bases de datos sincronizadas en cada una de las máquinas virtuales donde reside MEIWEB.

Para solucionar este problema se decidió implementar un sistema de base de datos homogéneos distribuidos, que es un conjunto de datos almacenados de manera sincronizada y con alta disponibilidad. La diferencia de este sistema, con el sistema centralizado que se utiliza actualmente consiste en que estos datos están almacenados en distintos nodos, que en el caso del aula virtual MEIWEB son máquinas virtuales que integran un sistema y que tienen conexión entre sí. Debido a que la información está distribuida en nodos, los resultados en las consultas se pueden obtener de manera rápida, ágil y fiable. Asimismo, en caso de que alguno de los nodos falle, como todos tienen autonomía local, el resto de nodos activos continúa trabajando sin que se desactive el sistema.

Este trabajo de investigación busca hacer un prototipo de base de datos distribuida (Clúster), el cuál será sometido a diferentes pruebas para verificar su funcionalidad y eficiencia. Finalmente, de ser aprobado el prototipo se procederá a la implementación en el aula virtual MEIWEB.

*Trabajo de grado en la modalidad de investigación.

**Facultad de Ingeniería Fisicomecánicas. Escuela de Ingeniería de Sistemas e Informática. Director MSc. Manuel Guillermo Flórez Becerra.

ABSTRACT

TITLE: IMPLEMENTTION OF HOMOGENEUS DISTRIBUTED DATABASE IN THE VIRTUAL CLASSROOM MEIWEB.*

AUTHORS: DE LAS AGUAS CASTELLÓN, Daniel David**

ACEVEDO GUTIÉRREZ, Juan Carlos**

KEYWORDS: Distributed databases, MySQL Cluster.

DESCRIPTION: MEIWEB is a virtual classroom that provides academic services to the Industrial University of Santander (UIS) using cloud technology, which involves handling a large amount of data stored in a centralized database; in order to optimize and increase the reliability and high availability of virtual classroom is necessary the existence of multiple databases synchronized in each of the virtual machines which reside MEIWEB.

To solve this problem it was decided to implement a homogeneous distributed databases system, which is a set of stored data synchronously with high availability. The difference of this system with the centralized system currently used is that these data are stored on different nodes in the case of the virtual classroom MEIWEB are virtual machines that make up a system and have connection with each other. Because the information is distributed nodes, query results can be obtained fast, flexible and reliable manner. Also, if any of the nodes fail, as all have local autonomy, the remaining active nodes continues to work without the system is turned off.

This research seeks to do a distributed database prototype (clúster), which will undergo different tests to verify functionality and efficiency. Finally, if approved prototype proceed to implementation in the virtual classroom MEIWEB.

* Degree Work in the form of Research.

**Physical-Mechanical Faculty of Engineering. School of Systems Engineering and Informatics. Director MSc. Manuel Guillermo Flórez Becerra.

INTRODUCCIÓN

Una base de datos distribuida (BDD) es una serie de múltiples bases de datos relacionadas entre sí, las cuales se encuentran distribuidas en diferentes espacios lógicos e intercomunicados mediante una red de comunicaciones, tienen total autonomía para realizar procesamientos locales y globales.

Un sistema de bases de datos distribuidas consiste en nodos con acoplamiento sutil y no comparten componente físico alguno. Es posible que los Sistemas de BD que son ejecutados en los distintos nodos tengan un grado sustancial de independencia mutua.

Entre las ventajas de utilizar bases de datos distribuidas encontramos una mayor tolerancia a fallos, lo que significa mayor disponibilidad, mejor rendimiento, además de autonomía local. Sin embargo, la principal ventaja de este tipo de bases de datos es que simplifican las consultas a la base de datos.

Desde la aparición de los sistemas de bases de datos distribuidas hace varios años, parece que el cambio de los sistemas centralizados a los distribuidos a escala comercial está llegando.
Las bases de datos distribuidas son cada vez más usadas por las empresas y suponen una ventaja competitiva frente a los sistemas centralizados, siempre y cuando la empresa en cuestión tenga necesidad de usar una base de datos de este tipo. Lo más habitual es disponer de varias sedes y tener que manejar información común, para lo cual las bases de datos distribuidas son especialmente útiles.

El presente trabajo se enfoca en realizar un prototipo de base de datos distribuida que mejore el funcionamiento del aula virtual MEIWEB.

1. INFORMACIÓN DEL PROYECTO

1.1 DESCRIPCIÓN DEL PROBLEMA

Actualmente MEIWEB es un aula virtual que presta servicios académicos a la Universidad Industrial de Santander usando tecnología en la nube, lo cual conlleva al manejo de una gran cantidad de datos almacenados en una base de datos centralizada; con el objeto de optimizar y aumentar la confiabilidad del MEIWEB se hace necesario la existencia de varias bases de datos sincronizadas en cada una de las máquinas virtuales donde reside MEIWEB.

1.2 JUSTIFICACIÓN

Para solucionar este problema se decidió implementar un sistema de base de datos homogéneos distribuidos que es un conjunto de datos almacenados de manera sincronizada y con alta disponibilidad. La diferencia de este sistema consiste en que estos datos están almacenados en distintos nodos, que en el caso del aula virtual MEIWEB son máquinas virtuales que integran un sistema y que tienen conexión entre sí. Debido a que la información está distribuida en nodos, los resultados en las consultas se pueden obtener de manera rápida, ágil y fiable. Asimismo, en caso de que alguno de los nodos falle, como todos tienen autonomía local, el resto de nodos activos continúa trabajando sin que se desactive el sistema.

2. OBJETIVOS

2.1 OBJETIVO GENERAL

- Migrar la base de datos centralizada del aula virtual MEIWEB a una base de datos MySQL distribuida.

2.2 OBJETIVOS ESPECÍFICOS

- Realizar un estudio del estado del arte de las bases de datos distribuidas de MySQL.
- Realizar pruebas de funcionamiento con la base de datos distribuida, usando un prototipo en máquinas virtuales creadas en el servidor del grupo de investigación GID-CONUSS.
- Definir una estrategia de migración de la base de datos centralizada del aula virtual MEIWEB.
- Definir el diseño de la base de datos distribuida.
- Implementar la migración de la base de datos centralizada y realizar pruebas de funcionamiento con la base de datos distribuida en el aula virtual MEIWEB
- Proponer para su publicación un artículo científico acerca de los estudios realizados sobre la implementación de bases de datos MySQL distribuidas al aula virtual MEIWEB para luego ser publicado en una revista científica.

3. MARCO TEÓRICO

3.1 BASES DE DATOS

El concepto de una base de datos varía de acuerdo al contexto en que se utilice en el campo de la informática y especialmente en nuestro proyecto definiremos una **base de datos (BD)**, como un conjunto de datos organizados y relacionados entre sí, los cuales son almacenados y utilizados por los sistemas de información de una empresa o entidad en particular.
Entre las características más importantes de una BD encontramos: independencia lógica y física de los datos, redundancia mínima, integridad de los datos, seguridad de acceso, auditoría, copias de seguridad, entre otras.

3.1.1 Sistema gestor de bases de datos

Un **sistema gestor de bases de datos (SGBD)** consiste en un conjunto de datos interrelacionados y de programas para acceder a dichos datos. El principal objetivo de un SGBD es brindar mecanismos para almacenar y recuperar la información de una base de datos de manera tal que sea práctica y eficiente.

La gestión de los datos conlleva tanto la definición de estructuras para almacenar los datos, como la provisión de mecanismos para la manipulación de información. Además, los sistemas de bases de datos deben proporcionar la fiabilidad de la información almacenada, a pesar de los fallos y caídas del sistema o los intentos de acceso sin autorización. Si los datos van a ser compartidos entre múltiples usuarios, el sistema debe evitar posibles inconsistencias.

Existen varios tipos de bases de datos; el presente proyecto se enfocará en las **bases de datos distribuidas (BDD),** definidas de la siguiente manera:

3.2 BASES DE DATOS DISTRIBUIDAS

Una BDD es una serie de múltiples bases de datos relacionadas entre sí, las cuales se encuentran distribuidas en diferentes espacios lógicos e intercomunicados mediante una red de comunicaciones, tienen total autonomía para realizar procesamientos locales y globales.

3.2.1 Sistema de bases de datos distribuidas

Un sistema de bases de datos distribuidas consiste en nodos con acoplamiento sutil y no comparten componente físico alguno. Es posible que los SBD que son ejecutados en los distintos nodos tengan un grado sustancial de independencia mutua.

Las bases de datos son almacenadas en un sistema distribuido de bases de datos, quien a su vez se encuentra en varios computadores. Los medios de comunicación (redes de alta velocidad o las líneas telefónicas), son los que hacen posible el contacto entre los diferentes computadores de un sistema distribuido. Tanto los discos, como las memorias no son compartidos. El tamaño de los computadores de un sistema distribuido puede variar en tamaño y en función abarcando desde las estaciones de trabajo, hasta los sistemas más complejos y grandes.

Los términos "nodo" y "sitio" son los computadores que pertenecen al sistema distribuido, pero se usan dependiendo del contexto en el cual se esté hablando.

"Un sistema gestor de bases de datos (SGBD) consiste en una colección de datos interrelacionados y una colección de programas para acceder a esos datos. Los datos describen una empresa particular. El objetivo principal de un SGBD es proporcionar un entorno que sea tanto conveniente como eficiente para las personas que lo usan para la recuperación y almacenamiento de la información"[1]

En un sistema distribuido se utilizan dos tipos de transacciones; las locales y las globales. "Una **transacción local** es aquella que accede a los datos del único sitio en el cual se inició la transacción. Por otra parte, una **transacción global** es aquella que, o bien accede a los datos situados en un sitio diferente de aquel en el que se inició la transacción, o bien accede a datos de varios sitios distintos."[2]

Existen algunas ventajas de usar sistemas distribuidos de bases de datos, como por ejemplo el compartimiento de datos, la autonomía y la disponibilidad, las cuales se explican a continuación:

[1] Abraham Silberschatz, Henry F. Korth, S. Sudarshan, "Fundamentos de bases de datos", cuarta edición, McGRAW-HILL/INTERAMERICANA DE ESPAÑA, S. A. U., 2002. Página 14.
[2] Abraham Silberschatz, Henry F. Korth, S. Sudarshan, "Fundamentos de bases de datos", cuarta edición, McGRAW-HILL/INTERAMERICANA DE ESPAÑA, S. A. U., 2002. Página 455.

Datos compartidos:

Capacidad para disponer de un entorno donde los usuarios puedan acceder desde una sola ubicación a los datos que residen en otras ubicaciones. Por ejemplo, en un sistema de banca distribuida, donde cada sucursal almacena datos relacionados con dicha sucursal, es posible que un usuario de una de las sucursales acceda a los datos de otra sucursal. Sin esta capacidad, un usuario que quisiera transferir fondos de una sucursal a otra tendría que recurrir a algún mecanismo externo que pudiera enlazar los sistemas existentes.

Autonomía:

Cada ubicación es capaz de mantener un grado de control sobre los datos que se almacenan localmente. En un sistema distribuido, existe un administrador de bases de datos global encargado de todo el sistema. Algunas responsabilidades del administrador local se delegan al administrador de bases de datos local de cada nodo. Dependiendo del diseño del sistema distribuido de bases de datos, cada administrador puede tener un grado diferente de autonomía local. La posibilidad de autonomía local es a menudo una de las grandes ventajas de las bases de datos distribuidas.

Disponibilidad:

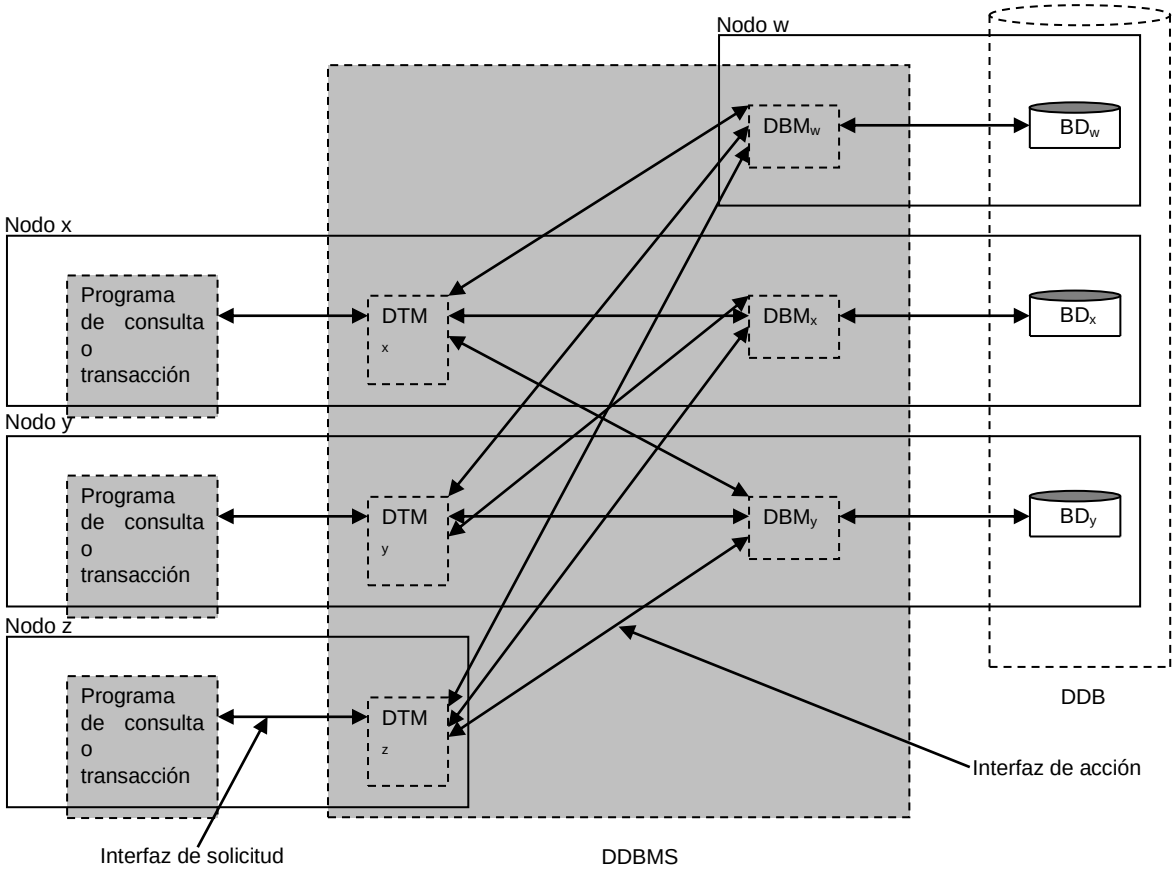
Si un sitio de un sistema distribuido falla, los sitios restantes pueden seguir funcionado. En particular, si los elementos de datos están replicados en varios nodos una transacción que necesite un elemento de datos en particular puede encontrarlo en otros nodos. De esta manera, el fallo de un nodo no implica necesariamente la caída del sistema. El sistema puede detectar el fallo de un nodo, y pueden ser necesarias acciones apropiadas para recuperarse del fallo. El sistema no debe seguir utilizando los servicios del nodo que falló. Finalmente, cuando el nodo que estaba caído se recupera, debe haber mecanismos disponibles para integrarlo sin problemas de nuevo al sistema.

Aunque la recuperación ante un fallo es más compleja en los sistemas distribuidos que en los sistemas centralizados, la capacidad que tienen muchos sistemas de continuar trabajando a pesar del fallo en uno de los nodos produce mayor disponibilidad. La disponibilidad es crucial para los sistemas de bases de datos que se utilizan en aplicaciones de tiempo real.

Los **sistemas gestores de bases de datos distribuidas (SGBDD)** constan sólo de una base de datos lógica que lee y procesa información de tablas, que se encuentra distribuida o dividida en distintos sitios que están interconectados mediante una red de comunicaciones, de esta manera existirán aplicaciones locales que sólo accederán a los datos que se encuentren almacenados en el mismo sitio, pero además habrá aplicaciones globales que accederán a datos desde varios sitios y así, el SGBDD llevará a cabo las funciones necesarias para dar servicio a este tipo de aplicaciones.

El procesamiento de bases de datos distribuidas, es el procesamiento de bases de datos en el cual la ejecución de transacciones, la recuperación y actualización de la información se lleva a cabo por medio de dos o más computadoras independientes y separadas geográficamente. La figura 1 muestra un sistema de base de datos distribuida que involucra cuatro computadoras.

Figura 1. Arquitectura de base de datos distribuida.



Fuente: Arquitectura de bases de datos distribuida. Departamento de Ingeniería de Sistemas, maestría en ciencias de la computación, Universidad de Colima, "Sistemas de bases de datos distribuidas". Página 1.

El **sistema de administración de base de datos distribuida** (DDBMS) está formado por las transacciones y los administradores de base de datos distribuidos de todas las computadoras. Tal y como se muestra, el DDBMS es un esquema genérico que implica un conjunto de programas que operan en diversas computadoras. Estos programas pueden ser subsistemas de un producto único DDBMS proporcionados por un sólo fabricante, o también pudiera resultar una colección de programas de fuentes dispares: algunos proporcionados por fabricantes, y algunos otros escritos en casa. El propósito de esta figura es ilustrar las funciones que deban atenderse en el procesamiento de bases de datos distribuidas.

Un **administrador de transacciones distribuidas** (DTM) es un programa que se encarga de administrar las solicitudes de procesamiento de los programas de consulta o de transacciones y también las ejecuta para los DDBM. Dependiendo de la naturaleza de la aplicación del administrador de bases datos distribuidas, el DTM puede ser dado como parte de DDBMS. En aplicaciones de baja complejidad, sus funciones pueden ser realizadas por personas.

El **administrador de la base de datos** (DBM) es un software que se encarga de procesar la base de datos distribuida (recuperar y actualizar datos del usuario) basándose en comandos de ejecución recibidos de los administradores de bases de datos. El DBM o gestor de bases de datos puede ser una derivada de un producto DDBMS, y su vez un sistema administrador de bases de datos no distribuido de uso comercial. En algunos casos, el DDBMS pudiera contener diferentes productos DBMS. Los DBMS pueden ser contenidos, en algunos casos, como productos de DDBMS.

Un **nodo o sitio** es una computadora con capacidad de ejecutar un DTM, un DBM, o ambos dependiendo del caso. Los DTM son procesados por nodos o sitios de transacción y a su vez, la base de datos y el DMB son procesados por el nodo de la base de datos.

3.3 VENTAJAS Y DESVENTAJAS DE LAS BASES DE DATOS DISTRIBUIDAS

3.3.1 Ventajas

Algunas de las ventajas más importantes de usar sistemas de bases de datos distribuidas son:

Mejor Rendimiento

Los datos se pueden almacenar cerca del lugar que los vaya a utilizar para que así, el tiempo de comunicación sea lo más corto posible. Si se tienen varios computadores funcionando de

manera simultánea, la cantidad de datos procesados será mayor que cuando sólo se tiene un solo computador.

Mayor Confiabilidad: La replicación de los datos aumenta su confiabilidad. Cuando falla una computadora o nodo, es posible la recuperación de los datos extraídos de otros nodos. Los usuarios no dependen de la disponibilidad de una sola fuente para sus datos.

Tamaño variable: Se pueden agregar o eliminar nodos o computadoras adicionales a la red conforme aumentan el número de usuarios y su carga de procesamiento.

3.3.2 Desventajas

Algunas de las desventajas que se pueden destacar acerca del procesamiento de bases de datos distribuidas, es que hay mayor complejidad en comparación con sistemas de bases de datos centralizados, lo que frecuentemente significa altos gastos de desarrollo y mantenimiento. El control de concurrencia y recuperación de fallas puede llegar a ser algo complicado y difícil de implementar, puede generar a una carga más grande en programación, los costos son mayores en cuanto a tiempo se refiere.

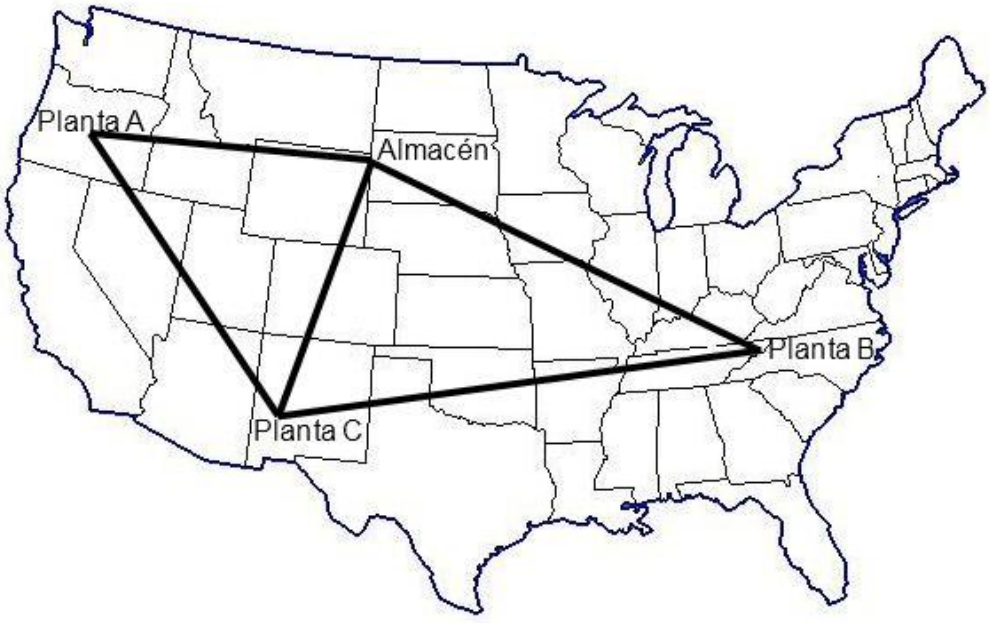
El procesamiento de bases de datos distribuido es difícil de controlar. Una computadora centralizada reside en un entorno controlado, con personal de operaciones que supervisa muy de cerca, y las actividades de procesamiento pueden ser vigiladas, aunque a veces con dificultad. En un sistema distribuido, las computadoras de proceso residen muchas veces en las áreas de trabajo de los usuarios. En ocasiones el acceso físico no está controlado, y los procedimientos operativos son demasiado suaves y efectuados por personas que tienen escasa apreciación o comprensión sobre su importancia. En sistemas centralizados, en caso de un desastre o catástrofe, la recuperación puede ser más difícil de sincronizar.

Ventajas y desventajas del procesamiento de una base de datos distribuida

Tabla 1. Ventajas y desventajas de las bases de datos distribuidas.

VENTAJAS	DESVENTAJAS
Mayor rendimiento	Difícil control de concurrencias
Mayor confiabilidad	Aseguramiento de la integridad de los datos.
Cambia de tamaño con facilidad	Mayor complejidad
Mayor tolerancia a fallos	Costos más altos
Control local de los datos	Difícil de controlar

Figura 2. Un negocio distribuido geográficamente



Fuente: Un negocio distribuido geográficamente. Departamento de Ingeniería de Sistemas, maestría en ciencias de la computación, Universidad de Colima, "Sistemas de bases de datos distribuidas". Página 2.

"Por último, los sistemas distribuidos se pueden adecuar de una manera más sencilla a las estructuras de la organización de los usuarios. La Figura 2 muestra la organización de un fabricante distribuido geográficamente. Los gerentes generales de cada una de las plantas poseen una enorme autoridad y libertad en la operación de sus instalaciones. Si tales plantas dependieran de una computadora única centralizada, la arquitectura de sistema entraría en conflicto con la filosofía y las políticas operacionales de la empresa. Incluso en organizaciones más centralizadas, el procesamiento distribuido ofrece una mayor flexibilidad para adecuarse a la estructura organizacional, de lo que permite el procesamiento centralizado. "[3]

3.4 TIPOS DE BASES DE DATOS DISTRIBUIDAS

Dentro de las bases de datos distribuidas encontramos dos tipos: Las homogéneas y las heterogéneas. En el caso de estudio se utilizarán bases de datos distribuidas homogéneas ya que la base de datos de la plataforma MEIWEB es homogénea.

En las **bases de datos distribuidas homogéneas** los sitios tienen un software de sistemas gestores de bases de datos de características similares saben de la existencia de los demás sitios y están programados para responder entre todos las solicitudes de los usuarios. En este tipo sistemas los sitios locales dejan a un lado una parte de su autonomía, es decir, no pueden modificar los esquemas o el SGBD. En este software, los sitios deben trabajar de manera conjunta para realizar el intercambio de la información sobre las transacciones, de esta manera se hace posible el procesamiento de dichas transacciones entre variedad de sitios.

A diferencia de lo anterior, en las **bases de datos distribuidas heterogéneas,** los sitios pueden trabajar con esquemas diferentes y diferentes SGBD. Es posible que algunos sitios no sepan de la existencia de los demás y así solo proporcionan facilidades limitadas para la cooperación en el procesamiento de las transacciones. El procesamiento de consultas puede estar inmerso en problemas debido a las diferencias en los esquemas mientras que la divergencia del software

[3] Un negocio distribuido geográficamente. Departamento de Ingeniería de Sistemas, maestría en ciencias de la computación, Universidad de Colima, "Sistemas de bases de datos distribuidas". Página 2.

puede llegar a ser un inconveniente para el procesamiento de transacciones que tengan acceso a varios nodos.

Hoy en día existen distintos productos comerciales con bases de datos distribuidas de varias empresas reconocidas, tales como: Relational Technology Inc, Oracle Corp e IBM.
Por el lado de la academia también tenemos software, entre ellos tenemos: Microsoft Access de Microsoft Corp, Oracle9i y MySQL. Este último recientemente lanzó su edición para BDD.
Cada uno de los anteriores tiene sus ventajas y sus desventajas.
En el caso de Microsoft Access, esta cuenta con un SGBD demasiado básico ya que está orientado a bases de datos muy poco sofisticadas, tampoco tiene en funcionamiento un SGBDD, pero tiene la ventaja de que se pueden implementar módulos que simulen la fragmentación, replicación y consultas distribuidas pero sin garantizar la fiabilidad que daría un SGBDD.
Oracle9i es más tiene un SGBD más escalable que Microsoft Access y también permite crear módulos que simulen fragmentación, replicación y consultas distribuidas. No tiene un SGBDD.
MySQL ya cuenta con un SGBDD.
En bases de datos tradicionales, la redundancia se fue reduciendo por dos razones:

- La inconsistencia entre las diferentes copias de los mismos datos lógicos son automáticamente anuladas por tener sólo una copia.
- El espacio es salvado por la eliminación de la redundancia.

El grupo de investigación GID-CONUSS es el encargado del mantenimiento y optimización de la plataforma cloud MEIWEB, la cual maneja un sistema de bases de datos centralizada. MEIWEB funciona en la nube, basándose en sistema de máquinas virtuales, y en una de ellas se encuentra almacenada la base de datos, que a la vez, se encuentra conectada por medio de nodos a las otras máquinas virtuales, lo que implica que al generarse un fallo en la máquina virtual principal, las otras no pueden acceder a la base de datos y MEIWEB deja de funcionar. Esta es la principal razón que nos hace querer implementar un sistema de bases de datos distribuido.

3.5 CLÚSTER DE BASES DE DATOS

Un clúster es un conjunto de máquinas que trabajan como como unidad que funcionan en conjunto para solucionar una única tarea, ahora bien, un MySQL Cluster hace referencia a un grupo de máquinas de trabajo en conjunto usando el motor de almacenamiento NDB (“Network Database”) de MySQL para implementar almacenamiento de datos, recuperación y

administración distribuida entre varias máquinas, lo que nos permite soportar una base de datos MySQL distribuida en una arquitectura de compartición nula usando almacenamiento en memoria.

Para que un clúster funcione correctamente, no es suficiente con conectar entre sí las máquinas, además es necesario configurar un sistema de manejo del clúster, el cual se encargue de interactuar con el usuario y los procesos que corren en él para optimizar el funcionamiento.[4]

Lo sistemas operativos en que MySQL Cluster está disponible son Linux, Mac OS X, Solaris y Windows (Hasta el momento de realización de este proyecto).

Para este proyecto se utilizó en cada una de las máquinas virtuales, el sistema operativo debian Wheezy 7.7.0 con versión de kernel 3.2.0-4-486.

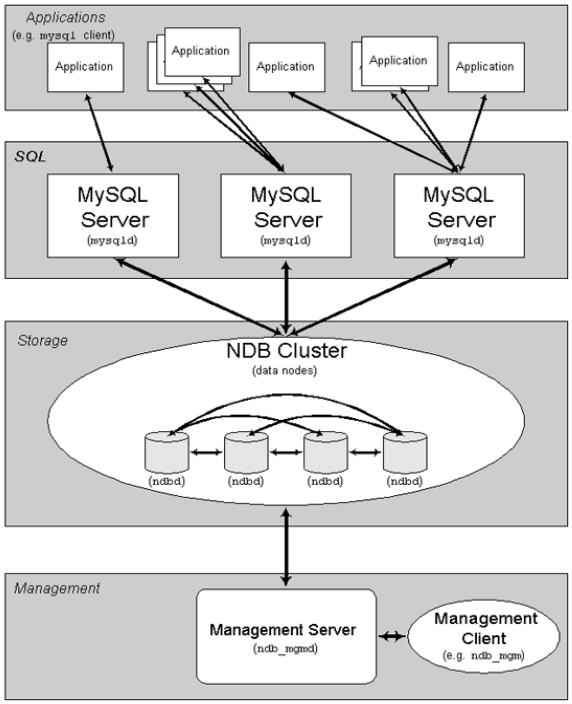
MySQL Cluster es un software que ayuda a la distribución de bases de datos en memoria en un entorno en donde los datos no son compartidos. Esta arquitectura en donde no hay compartición permite que el sistema funcione con hardware de bajas especificaciones.

Un MySQL Cluster consiste en un conjunto de máquinas, cada una ejecutando un número de procesos incluyendo servidores MySQL, nodos de datos para ndbcluster, servidores de administración, y (posiblemente) programas especializados de acceso a datos. La relación de estos componentes en un clúster se muestra aquí: [5]

[4] Tomado de: http://tavoberry.com/blog/mysql-cluster/

[5] Tomado de: http://manuales.guebs.com/mysql-5.0/ndbcluster.html#multi-config

Figura 3. Panorámica de MySQL Cluster



Fuente: Guebs [En línea] Disponible en: http://manuales.guebs.com/mysql-5.0/ndbcluster.html#mysql-cluster-overview

Todos estos programas funcionan juntos para formar un MySQL Cluster. Cuando se guardan los datos en el motor NDB Cluster, las tablas se guardan en los nodos de datos. Tales tablas son automáticamente accesibles desde todos los otros servidores MySQL en el clúster. Por ello, en una aplicación de compras que almacene datos en un clúster, si se actualiza el precio de un producto, todos los otros servidores MySQL que acceden a estos datos pueden ver el cambio inmediatamente.

Los datos almacenados en los nodos de datos de MySQL Cluster pueden replicarse: el clúster puede tratar fallos de nodos de datos individuales sin otro impacto aparte de abortar unas pocas transacciones debido a la pérdida de estado de transacción. Como las aplicaciones transaccionales se suponen que tratan fallos transaccionales, esto no debería ser un problema.

3.5.1 Conceptos básicos de MySQL Cluster

NDB: Motor de almacenamiento en memoria que ofrece alta disponibilidad y características de persistencia de datos.
El motor NDB puede configurarse con un rango de opciones de fallo y balanceo de carga, pero es más sencillo iniciarlo con el motor de almacenamiento a nivel de clúster. El motor de MySQL Cluster NDB contiene un conjunto completo de datos, dependiente sólo de otros datos dentro del propio clúster.

Nodo: Es considerado como nodo, cada una de las partes de un MySQL Cluster. Aquí a diferencia de otros contextos, el término "nodo" significa proceso, por lo tanto puede haber varios nodos en una misma máquina.

Existen tres tipos de nodos clúster, y en una configuración MySQL Cluster básica, mínimo habrá tres nodos, uno de cada tipo:

El nodo de administración (MGM): Este nodo es el encargado de administrar los otros nodos dentro del clúster, proporcionar datos de configuración, hacer copias de seguridad, arrancar y detener nodos. El nodo administrador debe ser el primero en ser iniciado. Un nodo MGM se arranca con el comando *ndb_mgmd*.
El nodo de datos: Es el encargado de almacenar los datos del clúster. En este proyecto se utilizaron dos nodos de este tipo. Un nodo de datos se arranca con el comando *ndbd*.
El nodo SQL: Este es el nodo que accede a los datos del clúster. En el caso de MySQL Cluster, un nodo cliente es un servidor MySQL tradicional que usa el motor ndbcluster. Un nodo SQL normalmente se arranca con el comando *mysqld --ndbcluster* o simplemente usando *mysqld* con *ndbcluster* añadido a *my.cnf.*

La configuración de un clúster consiste en configurar cada nodo en el clúster e inicializar los medios de comunicación entre nodos. En MySQL Cluster los nodos de datos son configurados homogéneamente en términos de procesador, espacio de memoria, y ancho de banda. Además, para asegurar un punto único de configuración, todos los datos de configuración del clúster entero se guardan en un único archivo de configuración.

El nodo MGM administra el archivo de configuración del clúster y el log. Cada nodo en el clúster recibe los datos de configuración del nodo de administración, y necesita una forma de

determinar dónde reside el nodo MGM. Cuando se presentan cambios en los nodos de almacenamiento de datos, los nodos transfieren información acerca de estos cambios al nodo de administración, que guarda la información en el log del clúster.
Además, puede haber cualquier número de procesos clientes del clúster o aplicaciones. Existen dos tipos:

Clientes MySQL estándar: MySQL Cluster permite el acceso a aplicaciones MySQL existentes escritas en C, C++, Java, Perl, PHP, Python, Ruby, entre otros lenguajes de programación.
Clientes de administración: Estos clientes se conectan al nodo de administración (nodo MGM) y son los encargados de arrancar y parar nodos, arrancar y parar traceo de mensajes (sólo en versiones de depuración), mostrar versiones y estado de los nodos, arrancar y parar copias de seguridad, entre otras acciones.

3.5.1.1Mysqld

El proceso de servidor de bases de datos tradicional que puede ser utilizado en ambientes de clúster o aislado. Para que este proceso sea utilizado dentro de un clúster MySQL, necesita ser construido especialmente para soportar el motor de almacenamiento NDB, los binarios compilados disponibles en el sitio de MySQL ya integran esta funcionalidad al proceso.
Una manera fácil de asegurarse que se dispone de la versión correcta para un clúster MySQL es invocando el comando SHOW ENGINES dentro del ambiente desde el servidor buscando la existencia del motor NDB. Este comando muestra la totalidad de los motores soportados por el proceso actualmente instalado.

Para poder unir un servidor MySQL a un clúster, es necesario poder interactuar con el nodo de administración de dicho clúster. Para esto en el archivo de configuración de MySQL (my.cfg) se debe especificar el string de conexión a dicho servidor. La comunicación entre servidores se realiza mediante el protocolo TCP/IP por lo cual es necesario indicar en el string de conexión la dirección IP del nodo administrativo y el puerto en el cual se publica el servicio de administración.

3.5.1.2Ndbd

El proceso *ndbd* es el encargado de manejar todos los datos de las tablas utilizando el motor *ndbd* clúster. Este proceso soporta la funcionalidad de manejo de transacciones distribuidas

entre los nodos, recuperación de nodos defectuosos o fuera de línea, *checkpoint to disk* (el momento en el que los datos son escritos efectivamente a disco), respaldo en línea, y otras tareas relativas a la distribución del clúster.

El conjunto de procesos distribuido *ndbd* cooperan colectivamente en la tarea de manejo de datos. Cada uno de estos procesos genera un conjunto de archivos de log independientes que es almacenado en el directorio *DataDir* especificado en el archivo de configuración de cada nodo de datos (config.ini). Para poder conectar un nodo de datos es necesario proveer al proceso *ndbd* con la información necesaria acerca del nodo administrativo. Análogamente a como se realiza con el nodo MySQL server, se debe especificar dirección IP y puerto del mismo en el archivo de configuración.

Cuándo un proceso *ndbd* comienza su ejecución, se levantan dos procesos, uno de ellos es llamado angel. El proceso angel supervisa la ejecución del proceso de almacenamiento, y en caso de falla, vuelve a iniciarlo. Por esto, si se intenta detener el proceso *ndbd* utilizando el comando *kill* de Unix, es necesario acabar con los dos procesos, comenzando con el proceso *angel*, dado que sino este volverá a iniciar el proceso de datos. La mejor manera de acabar con un proceso *ndbd* de un nodo es utilizando los comandos apropiados en el nodo de administración o utilizando un cliente de administración externo.

El proceso de ejecución utiliza un hilo para leer, escribir, escanear datos y realizar otras actividades. Este hilo está implementado de manera asíncrona con el objetivo de poder realizar fácilmente múltiples actividades concurrentes. Se utiliza otro hilo para supervisar que la ejecución no se detenga o genere un deadlock. El acceso a los archivos en el disco se realiza a través de múltiples hilos, manejando cada uno un archivo de datos en particular. De esta forma el proceso *ndbd* es capaz de hacer uso exhaustivo de arquitecturas con múltiples procesadores de una manera óptima.

3.5.1.3 Ndb_mgmd

Es el proceso que controla el servidor de administración, siendo responsable de conocer y mantener la configuración del clúster y distribuir dicha información a todos los nodos que la soliciten al unirse al clúster. Mantiene también el log de las actividades del clúster y reporta su estado a los clientes que se conectan a él.

En un esquema con un único servidor de administración, no es necesario especificar un string de conexión al nodo de administración dado que es el mismo el propio servidor. Si se está trabajando en un esquema donde existe más de un nodo de administración se debe especificar cada uno con un ID específico e indicar las cadenas de conexión a cada uno de ellos.

Junto con el proceso NDB_MGMD se encuentra *ndb_mgm*, que es responsable de manejar el cliente de administración que interactúa con el nodo de administración. El cliente de administración se comunica con el nodo de administración utilizando una API en C. Esta API se puede utilizar para desarrollar aplicaciones dedicadas a controlar y administrar el clúster de una manera personalizada.[6]

3.6 PROXY

Un proxy es aquel que intermedia en las peticiones que realiza un cliente A a un servidor C. Por ejemplo, si una máquina A solicita un recurso a C, lo hará mediante una petición a B, que a su vez trasladará la petición del recurso a C; de esta forma C no sabrá que la petición procedió originalmente de A. Se generan puntos intermedios se puede aprovechar para distintas funcionalidades, tales como: control de acceso, registro y denegación del tráfico, mejoras en rendimiento, anonimato, proporcionar caché web entre otras. La caché web sirve para acelerar y mejorar la experiencia del usuario mediante permisos que guardará la web, ya que la siguiente ocasión que se visiten las páginas web no se extraerá información de la web si no que se recuperará la información que fue almacenada en la caché.

Uno de los usos más frecuentes es el de servidor proxy para interceptar las conexiones de red que debe realizar un cliente para llegar a un servidor destino.
Un servidor proxy web Intercepta la navegación de los clientes por páginas web. Los motivos de su uso son varios y muy distintos, pueden ser: seguridad, rendimiento, anonimato.
Es el proxy quien realiza la comunicación en los casos en los que un equipo de la red desea acceder a una información o a cualquier recurso, para luego trasladar la respuesta al equipo desde el cual se hizo la solicitud.
Hay dos tipos de proxy dependiendo de quién es el que quiere implementar la funcionalidad del proxy; está el proxy local, al cual se le llama así porque el que quiere implementar la política es el mismo que hace la petición. Se encuentra en la misma máquina cliente que hace las

[6] Tomado de: http://es.wikipedia.org/wiki/MySQL_Cluster

solicitudes. Se usan en su mayoría para darle la opción al cliente de controlar el tráfico, así coma la de establecer reglas de filtrado, las cuales pueden evitar que se revela información privada. A diferencia de éste, el proxy externo se suelen usar para implementar cacheos, bloquear contenidos, control del tráfico, compartir IP; todo esto desde una entidad externa.

En redes de computadores, un proxy inverso es un tipo de servidor proxy que recupera los recursos en nombre de un cliente desde uno o más servidores. Esos recursos son entonces devueltos a el cliente como si hubiera sido originados desde el mismo servidor proxy. Mientras los proxy tradicionales actúan como intermediarios para sus clientes asociados, para luego ponerse en contacto con cualquier servidor, un servidor proxy inverso lo hace como un intermediario para servidores asociados, para luego ser contactados por cualquier cliente.

Esta son algunas de las razones más frecuentes para el uso de un proxy inverso.

- Seguridad: el servidor proxy da una capa más de defensa la cual protege a los servidores web.
- Cifrado / Aceleración SSL: cuando se crea un sitio web seguro, usualmente el cifrado SSL es realizado en un equipo ajeno equipado incluso con hardware de aceleración SSL/TLS.
- Balanceo de carga: el proxy puede distribuir la carga entre varios servidores web. En ese caso puede ser necesario reescribir la URL de cada página web (traducción de la URL externa a la URL interna correspondiente, según en qué servidor se encuentre la información solicitada).
- Caché de contenido estático: Un proxy inverso puede descargar de trabajo a los servidores web almacenando contenido estático como imágenes u otro contenido gráfico. También puede almacenar contenido generado dinámicamente pero que pueda ser en alguna medida reutilizable.

3.7 BALANCEADOR DE CARGAS

Un balanceador de carga básicamente es un dispositivo de hardware o software que se usa para que un grupo de servidores atiendan una aplicación asignándoles o balanceando las solicitudes que llegan de los clientes a dichos servidores usando algoritmos, como por ejemplo, un simple Round Robin hasta otros de mayor sofisticación.

3.8 VIRTUALBOX

Oracle VM VirtualBox es un software de virtualización para arquitecturas x86/amd64, propiedad de Oracle Corporation. Éste hace parte de sus productos de virtualización. Oracle VM VirtualBox permite instalar “sistemas invitados” ya que están alojados dentro del sistema operativo principal o “anfitrión”, cada uno funcionando de manera independiente, pero con la opción de comunicación por red.

Sistemas operativos soportados (modo anfitrión):

- GNU/Linux
- Mac OS X
- OS/2 Warp
- Microsoft Windows
- Solaris/OpenSolaris

Sistemas operativos soportados (modo invitado):

- FreeBSD
- GNU/Linux
- OpenBSD
- OS/2 Warp
- Windows
- Solaris
- MS-DOS y muchos otros.

El hardware emulado tiene los discos duros de los sistemas invitados almacenados en los sistemas anfitriones en un contenedor llamado “Virtual Disk Image” pero en forma de archivos individuales lo cual genera una incompatibilidad con el resto de software de virtualización.

3.9 HAPROXY

HAProxy es una solución de alta disponibilidad de tipo "open source" que proporciona equilibrio de carga y funciones proxy para aplicaciones basadas en TCP y HTTP mediante la difusión de las solicitudes a través de múltiples servidores.

Este es adecuado para sitios web de alto tráfico. Con el paso de los años HAProxy se ha convertido en el balanceador de carga de código abierto estándar, ahora trabaja con las principales distribuciones de Linux, y con frecuencia se despliega por defecto en plataforma en la nube

4. ESTADO DEL ARTE

4.1 TIPOS DE FRAGMENTACIÓN EN BASES DE DATOS DISTRIBUIDAS

Una base de datos distribuida (BDD) es una colección de datos que pertenecen lógicamente al mismo sistema, pero que están distribuidos sobre diferentes ordenadores de la red. Esta definición enfatiza dos aspectos importantes en una BDD:

Distribución: Los datos no residen en el mismo lugar. De este modo se puede distinguir una BDD de una base de datos (BD) centralizada.

Correlación lógica: Es decir, el hecho por el cual los datos tienen algunas propiedades o características que los relaciona. De este modo se puede distinguir una BDD de un conjunto de BD locales o ficheros residentes en diferentes lugares de una red de ordenadores.

En la arquitectura de una BDD se pueden identificar tres capas: vista de usuario, conceptual y física. En la capa de usuario se encuentran todas las aplicaciones, las pantallas de entrada de cada dato y los reportes requeridos para la empresa para la cual se representa el modelo. En la capa conceptual está el modelo del negocio subyacente, usualmente el modelo entidad-relación. En la capa física está el modelo físico y la estructura de la base de datos. Desde la perspectiva de una BDD, los usuarios desde la capa de usuario examinan y manipulan los datos como si la base de datos estuviese centralizada en ese nodo. Desde el punto de vista físico la BD está fragmentada y ubicada en diferentes sitios.

Entre los beneficios que presentan las bases de datos distribuidas, se encuentran:

- Los datos son localizados en un lugar más cercano, por tanto, el acceso es más rápido.
- El procesamiento es rápido debido a que varios nodos intervienen en el procesamiento de una carga de trabajo, nuevos nodos se pueden agregar fácil y rápidamente.
- La comunicación entre nodos se mejora y existe una autonomía e independencia entre los nodos.
- Los datos se pueden colocar físicamente en el lugar donde se acceden con mayor frecuencia, haciendo que los usuarios tengan control local de los datos con los que interactúan. Esto resulta en una autonomía local.

- Mediante la replicación de información, las bases de datos distribuidas pueden presentar cierto grado de tolerancia a fallas haciendo que el funcionamiento del sistema no dependa de un solo lugar.

4.1.1 Arquitectura de referencia para las bases de datos distribuidas

Los elementos que forman la arquitectura son los siguientes:

Esquema global: Define todos los datos que están contenidos en la BDD como si la BD no fuese distribuida. Por esta razón, el esquema global puede ser definido exactamente de la misma manera que una BD no distribuida. Refiriéndose a un modelo relacional, se tendrían relaciones globales.

Fragmentos: Cada relación global puede ser dividida en porciones que no se solapen, llamados fragmentos. El mapa resultante se denomina esquema de fragmentación. Una relación global puede dividirse en n fragmentos y un fragmento sólo puede pertenecer a una relación global. Los fragmentos se referencian por un nombre de relación global y un subíndice. Por ejemplo Ri, indica el fragmento i de la relación R.

- **FRAGMENTACIÓN**

Los principales problemas de la fragmentación se pueden resumir básicamente en:

Encontrar la unidad apropiada de distribución, es decir, definir qué contiene un fragmento.

El rendimiento se afecta cuando existen aplicaciones que necesitan tener una vista completa de un objeto o entidad (relación, en el modelo relacional) y está descompuesta en fragmentos ubicados físicamente en sitios distintos. Esta recuperación requiere la ejecución de operaciones de unión y combinación.

Se pierde el significado semántico del objeto o entidad al tenerse el concepto subdivido en fragmentos ubicados en diferentes sitios. Esto puede provocar, por ejemplo, complicaciones en la interpretación y verificación del modelo conceptual que representa a los requerimientos.

A pesar de estos inconvenientes, la fragmentación facilita el proceso concurrente de las transacciones y, por lo tanto, la recuperación de información. Con el fin de realizar una fragmentación adecuada es necesario proporcionar información que ayude a realizarla. Esta información normalmente debe ser proporcionada por el usuario y tiene que ver con cuatro

tipos: 1. Información sobre el significado de los datos; 2. Información sobre las aplicaciones que los usan; 3. Información acerca de la red de comunicaciones; 4. Información acerca de los sistemas de cómputo.

Existen tres tipos de fragmentación: 1. Fragmentación horizontal; 2. Fragmentación vertical; 3. Fragmentación mixta o híbrida.

- **FRAGMENTACIÓN HORIZONTAL**

La fragmentación horizontal se realiza sobre las tuplas de la relación. Cada fragmento será un subconjunto de las tuplas de la relación. Existen dos variantes de la fragmentación horizontal: la primaria y la derivada. La fragmentación horizontal primaria de una relación se desarrolla empleando los predicados definidos en esa relación. Por el contrario, la fragmentación horizontal derivada consiste en dividir una relación partiendo de los predicados definidos sobre alguna otra.

- **FRAGMENTACIÓN VERTICAL**

El objetivo de la fragmentación vertical es particionar una relación en un conjunto de relaciones más pequeñas de manera que varias de las aplicaciones de usuario se ejecutarán sobre un fragmento. En este contexto, una fragmentación óptima es aquella que produce un esquema de fragmentación que minimiza el tiempo de ejecución de las consultas de usuario.

La fragmentación vertical ha sido estudiada principalmente dentro del contexto de los sistemas de manejo de bases de datos centralizados como una herramienta de diseño, la cual permite que las consultas de usuario traten con relaciones más pequeñas haciendo, por tanto, un número menor de accesos a páginas. Es más complicada que el particionamiento horizontal, ya que existe un gran número de alternativas para realizarla.

- **FRAGMENTACIÓN MIXTA O HÍBRIDA**

La forma más simple de construir fragmentación mixta consiste en aplicar fragmentación horizontal a fragmentos verticales, o aplicar fragmentación vertical a fragmentos horizontales. Por ejemplo:

a) Fragmentación vertical seguida por una fragmentación horizontal

b) Fragmentación horizontal seguida por una fragmentación vertical

4.1.1 Alternativas sobre replicación para la asignación de fragmentos

Si la fragmentación es correcta se decide cómo asignar tales fragmentos a los diferentes sitios de la red. En el caso en que una serie o grupo de datos sean asignados para la fragmentación, pueden ser replicados para mantener una copia. Al existir varias copias de un elemento de datos cuando falle el sistema se podría tener acceso a esos datos, los cuales están ubicados en sitios distintos. Si hay consultas que pretenden acceder a los mismos datos se pueden ejecutar de forma paralela, debido a las copias que hay en diferentes sitios. La ejecución de consultas de actualización y de escritura implicaría la actualización de todas las copias que existan en la red, cuyo proceso puede resultar complicado y generar problemas.

Debido a lo anterior, un buen parámetro para afrontar el grado de réplica sería la cantidad de consultas de lectura que se efectuarán, así como el número de consultas de escritura que se llevarán a cabo. En una red donde las consultas que se procesen sean mayoritariamente de lectura, se podrá alcanzar un alto grado de réplica, no así en el caso contrario.

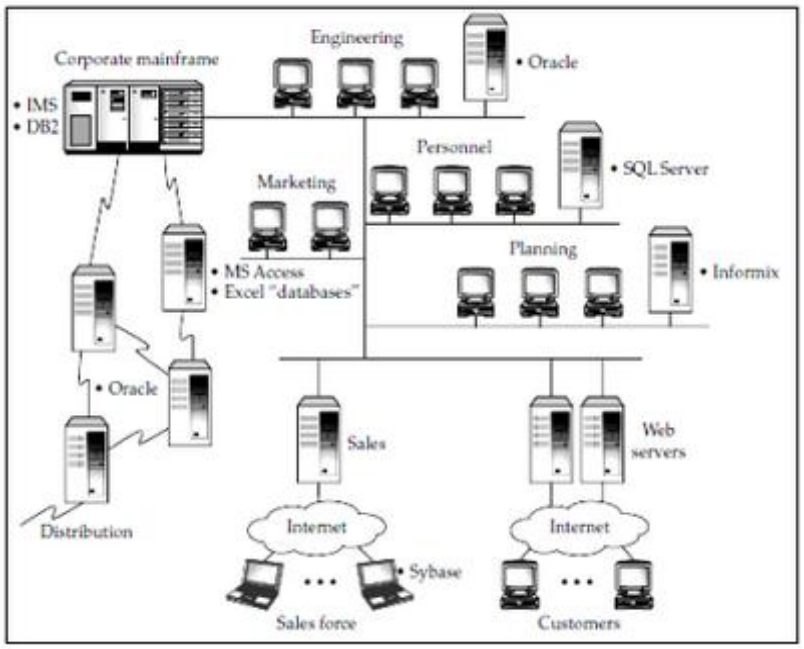
Una base de datos fragmentada es aquella donde no existe réplica alguna. Los fragmentos de la base de datos son almacenados en sitios en los cuales existe solamente una copia de cada uno de ellos en toda la red. En caso de réplica, se puede considerar una base de datos totalmente replicada, donde existe una copia de todo el banco de datos en cada sitio, o considerar una base de datos parcialmente replicada donde existan copias de los fragmentos ubicados en diferentes sitios.

4.2 GESTIÓN DE BASES DE DATOS DISTRIBUIDA USANDO MÉTODO DE REPLICACIÓN

4.2.1 Datos distribuidos

Cuando los fundamentos de la gestión de bases de datos relacionales y el lenguaje SQL se están sentando en la década de 1970, casi todo el procesamiento de datos comerciales sucedió en sistemas informático grandes y centralizados.

Figura 4. Uso de un DBMS en una red empresarial típica.

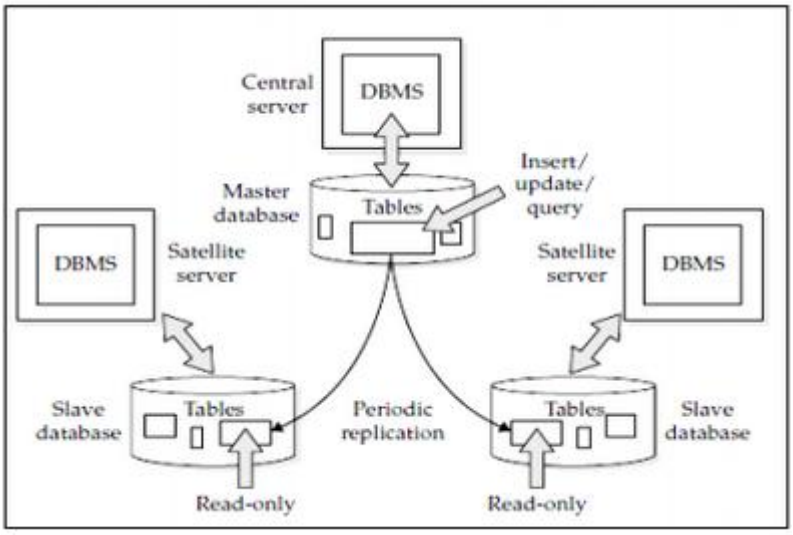


Fuente: Mircea petrini, Distributed Databases Management Using Replication Method.

4.2.2 Extracción de tablas

Cuando el acceso remoto crece más allá de un cierto punto, suele ser más eficiente para mantener una copia local de los datos remotos en la base de datos local. Los proveedores de DBMS proporcionan herramientas para simplificar el proceso de extracción y distribución de datos. El proceso extrae el contenido de una tabla en una base de datos maestra, lo envía a través de una red a otro sistema, y la carga en una tabla de réplica correspondiente en una base de datos esclavo, como se muestra en la figura 5. En la práctica, la extracción se realiza periódicamente y fuera de las horas de máxima actividad de base de datos.

Figura 5. Una arquitectura básica maestro/esclavo.



Fuente: Mircea petrini, Distributed Databases Management Using Replication Method.

El primer paso hacia la automatización de esta estrategia fue el desarrollo de programas de extracción de datos y carga de datos de alta velocidad. Estos programas de utilidad, que ofrecen los proveedores de DBMS, suelen utilizar propietario, técnicas de acceso de base de datos de nivel inferior para extraer los datos y cargar los datos mucho más rápidamente de lo que es posible a través de SQL SELECT e INSERT.

4.2.3 Replicación de tabla

Varios proveedores de DBMS han ido más allá de sus programas de extracción y de la utilidad de carga para ofrecer apoyo para la extracción de la tabla dentro del propio DBMS. Oracle, por ejemplo, dispone de vista materializada para crear automáticamente una copia local de una tabla remota. Una vista materializada es una vista que almacena realmente las filas definidas por la consulta incluida en la definición de vista. En su forma más simple, la tabla local es una réplica de sólo lectura de la tabla maestra remota que se carga cuando se define la vista.

Create a local replica of pricing information from the remote PRODUCTS table.

CREATE MATERIALIZED VIEW PRODPRICE

AS SELECT MFR_ID, PRODUCT_ID, PRICE

FROM PRODUCTS@REMOTE_LINK;

The CREATE MATERIALIZED VIEW statement also includes rather comprehensive facilities for specifying automatic refreshes. Here are some examples:

Create a local replica of pricing information from the remote PRODUCTS table. Refresh the data once per week, with a complete reload of the data.

CREATE MATERIALIZED VIEW PRODPRICE

REFRESH COMPLETE START WITH SYSDATE NEXT SYSDATE+7

AS SELECT MFR_ID, PRODUCT_ID, PRICE

FROM PRODUCTS@REMOTE_LINK;

Create a local replica of pricing information from the remote PRODUCTS table. Refresh the data once per day, sending only changes from the master table.

CREATE MATERIALIZED VIEW PRODPRICE

REFRESH FAST START WITH SYSDATE NEXT SYSDATE+1

AS SELECT MFR_ID, PRODUCT_ID, PRICE

FROM PRODUCTS@REMOTE_LINK;

De forma predeterminada, Oracle identifica filas (para determinar si se cambian) en función de su clave primaria. Si la clave principal no es parte de los datos replicados, esto puede causar confusión acerca de lo que se han actualizado las filas; en este caso, Oracle utiliza un número de fila-id interna (una opción que se puede especificar cuándo se crea la vista materializada) para identificar las filas modificadas por refresca a la vista materializada.

La sentencia *SELECT* que define la vista materializada ofrece una capacidad muy general para la extracción de datos. Puede incluir una cláusula *SELECT* para extraer sólo las filas seleccionadas de la tabla maestra:

Create a local replica of pricing information for high-priced products from the remote PRODUCTS table. Refresh the data once per day, sending only changes from the master table.

CREATE MATERIALIZED VIEW PRODPRICE

REFRESH FAST START WITH SYSDATE NEXT SYSDATE+1

AS SELECT MFR_ID, PRODUCT_ID, PRICE

FROM PRODUCTS@REMOTE_LINK WHERE PRICE > 1000.00;

Hay que tener en cuenta que los predicados en los que no afectan el registro de cambios. Todos los cambios en la tabla *PRODUCTOS* aún deben ser registrados porque múltiples vistas materializadas se pueden actualizar desde el registro de cambios, independientemente de los predicados utilizados en sus definiciones. La vista materializada también se puede crear como una tabla unida, extraer sus datos de dos o más tablas maestras en la base de datos remota:

Create a local replica of salesperson data, refreshed weekly.

CREATE MATERIALIZED VIEW SALESTEAM

REFRESH FAST START WITH SYSDATE NEXT SYSDATE+7

AS SELECT NAME, QUOTA, SALES, CITY

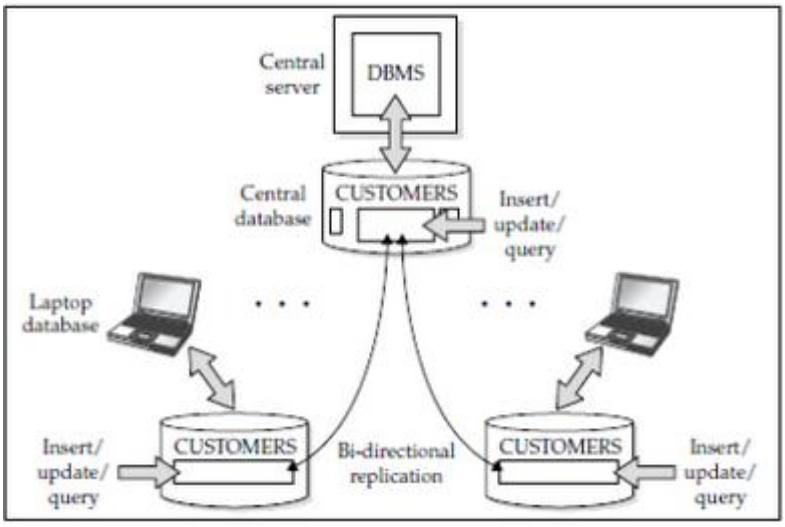
FROM SALESREPS@REMOTE, OFFICES@REMOTE WHERE REP_OFFICE = OFFICE;

4.2.4 Replicas actualizables

En las implementaciones más simples, una mesa y sus réplicas tienen una estricta relación maestro / esclavo, como se muestra en la figura 6. La copia central / maestro contiene los datos reales. Es siempre al día, y todos los cambios a la tabla deben ocurrir en esta copia de la tabla. Las otras copias de esclavos están pobladas de actualizaciones periódicas, gestionados por el DBMS.

Para algunas aplicaciones, la replicación de tablas es una técnica excelente y sin la relación maestro / esclavo. Por ejemplo, las aplicaciones que exigen el uso de alta disponibilidad replican tablas para mantener copias idénticas de los datos en dos sistemas informáticos diferentes. Si un sistema falla, el otro contiene los datos actuales y puede continuar su procesamiento. Una aplicación de Internet puede exigir tasas muy altas de acceso de base de datos, y lograr este escalabilidad mediante la replicación de una tabla muchas veces en diferentes sistemas informáticos y luego extiendo la carga de trabajo entre los sistemas.

Figura 6. Replicas con múltiples sitios de actualización.



Fuente: Mircea petrini, Distributed Databases Management Using Replication Method.

5. ANÁLISIS

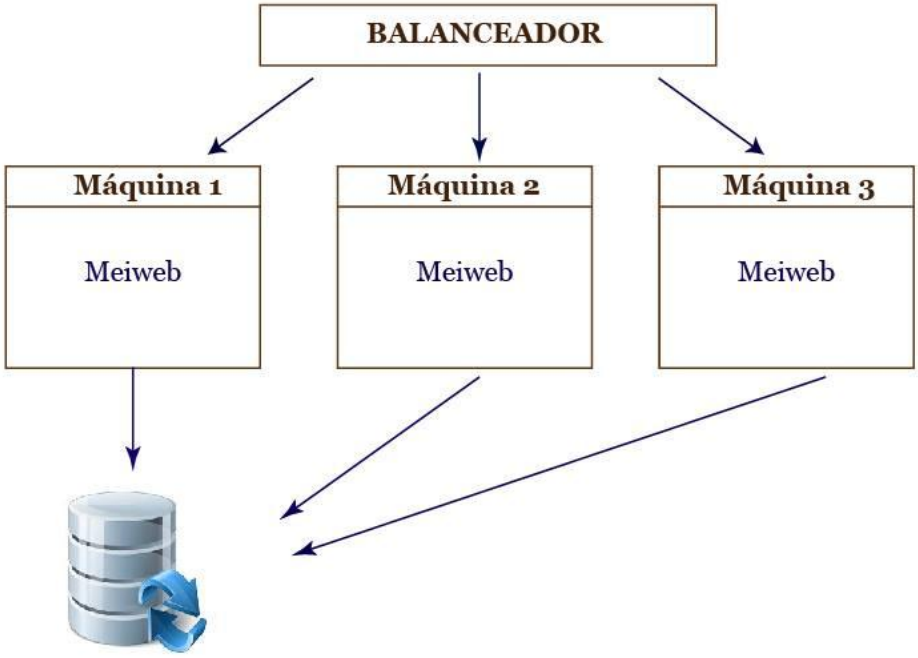
5.1 SITUACIÓN ACTUAL

El aula virtual MEIWEB es una plataforma a cargo del grupo de investigación GID-CONUSS, que le proporciona a los estudiantes y docentes de la escuela de Ingeniería de Sistemas e Informática de la Universidad Industrial de Santander un espacio para el desarrollo de actividades académicas, cursos virtuales y el manejo de asignaturas. MEIWEB permite a los docentes habilitar áreas de trabajo para sus asignaturas en las cuales los estudiantes pueden presentar parciales, subir trabajos, verificar sus notas, etc.

Actualmente está aula virtual en la nube funciona por medio de máquinas virtuales con sistema operativo Linux Debian y se encuentran organizadas de la siguiente manera:

Existen tres máquinas virtuales en las que se encuentra alojada la aplicación MEIWEB, estas máquinas cuentan con un balanceador de cargas que distribuye las peticiones de los usuarios para evitar caídas por congestión o por el fallo de alguna de las mismas. Dentro de dos de estas tres máquinas mencionadas se encuentran almacenados los datos guardados por los usuarios en un sistema de alta disponibilidad, que se actualiza conforme se generan transacciones proporcionando un acceso continuo a dichos datos. Cada una de estas máquinas apunta a la base de datos del aula, esta se encuentra almacenada en otra máquina virtual. A esto a es lo que se la conoce como una Base de datos Centralizada, ya que para cada uno de los nodos existe un único archivo de base de datos, lo que no proporciona una alta disponibilidad de la información requerida por los usuarios. Cuando se genera un fallo en la máquina virtual de datos ninguno de los nodos en los que se almacena la aplicación es capaz de acceder a los datos requeridos por el usuario, lo que genera un problema para el óptimo funcionamiento del sistema.

Figura 7. Situación actual MEIWEB.



5.2 PLANTEAMIENTO DE LA SOLUCIÓN

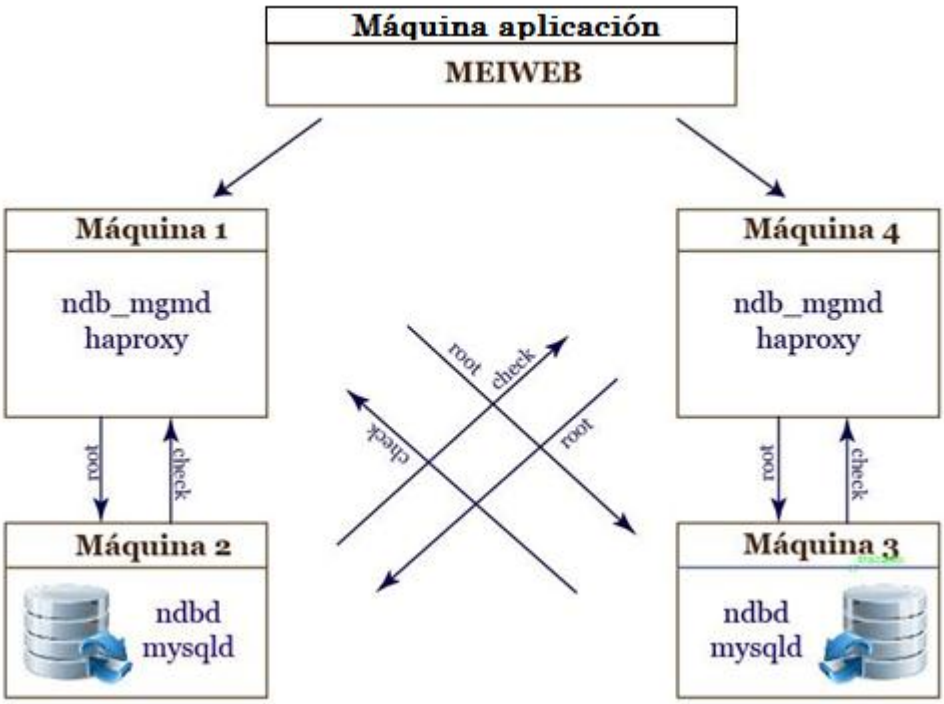
Los autores de este proyecto bajo la supervisión del grupo de investigación GID-CONUSS plantean como solución la implementación de un sistema de bases de datos distribuidas, para aumentar así la disponibilidad de los datos requeridos para el funcionamiento del aula virtual MEIWEB. El sistema de bases de datos distribuidas permite a los nodos tener un fragmento de la base de datos principal o como en el caso de estudio, una réplica de la misma.

El sistema de bases de datos distribuidos o clúster de bases de datos se diseñó de la siguiente manera teniendo en cuenta el estudio previo del sistema ya implementado, el cual aunque se encuentra en alta disponibilidad de datos no lo está en lo que a bases de datos respecta. Para el diseño del clúster se plantea, después de realizar un estudio del estado del arte, que no es necesario hacer fragmentación de la base de datos, ya que este tipo de distribución de datos sólo es necesario cuando, por ejemplo, una empresa se encuentra divida por departamentos y los datos a los cuales accede cada uno de ellos se encuentra en módulos específicos, por lo que se concluyó que el diseño actual es ideal, y para su mejoramiento sólo basta con hacer la replicación de la base de datos en el clúster de la siguiente manera:

Se utilizarán 4 máquinas virtuales en el clúster que se distribuyen en dos de administración del clúster y dos para el almacenamiento de los datos del mismo. En las máquinas o nodos administradores se encontrarán instalados el ndb_mgm y el balanceador de cargas, a su vez en las máquinas de datos estará mysql y el ndbd, el primero se encargará del almacenamiento de los datos, el segundo de su replicación. La aplicación de MEIWEB y sus datos se encontrarán alojadas en una quinta máquina virtual, la cual accederá en primera instancia a alguno de los nodos administradores. Se decide el uso de dos nodos administradores para que, en caso de algún fallo, la aplicación tenga la opción de seguir haciendo transacciones con la base de datos por medio del nodo administrador restante. Cuando la petición o transacción llega al nodo administrador este decide con ayuda del balanceador de cargas, a cual nodo de datos enviar la petición, revisando su disponibilidad en la red o la cantidad de peticiones en cola, todo esto por medio de un algoritmo ‘’Round Robin”.

Esta solución busca proveer mayor disponibilidad de los datos, más velocidad en las consultas y más confiabilidad en los datos obtenidos, además de reducir la posibilidad que tiene el sistema actual a tener fallas.

Figura 8. Diseño del clúster.



6. PROTOTIPO

6.1 CONFIGURACIÓN DEL CLÚSTER DE MYSQL

Para iniciar la configuración del clúster se debe instalar linux en máquinas virtuales, las cuales serán los nodos que conforman el clúster. Para este caso específico se utilizaron 4 máquinas virtuales con un distro debian Wheezy 7.7.0.

Como primera medida se debe crear una red interna para establecer comunicación entre los nodos del clúster. Para esto se debe ingresar al VirtualBox, seleccionar cada una de las máquinas e ir a "Configuración" y desde ahí seleccionar el ítem "Red". En esta parte hay varias opciones, se deben habilitar el adaptador de red 1 y el adaptador de red 2. El adaptador de red 1 se utiliza para acceder a la tarjeta de red el computador, y el adaptador 2 para habilitar la red interna.

Para empezar con la configuración del clúster de MySQL para 4 nodos: Dos nodos administradores ("maquina1" y "maquina4") y dos nodos de datos ("maquina2" y "maquina3"). Lo primero que se debe hacer es descargar el MySQL server en cada uno de los nodos. MySQL Server se compone de tres paquetes: *mysql-common*, *mysql-client* y *mysql-server*.

Después de hacer la instalación de *mysql server* se procede a configurar MySQL Cluster en cada uno de los nodos, empezando por los nodos administradores, para lo cual debemos descargar el paquete "mysql-cluster-gpl-7.4.6-linux-glibc2.5-i686.tar.gz" que es el genérico de linux desde la página http://dev.mysql.com/downloads/cluster/.

La carpeta de MySQL que fue descargada y descomprimida contiene archivos de tipo binario, los cuales son los encargados de iniciar los servicios en los nodos. Se debe copiar los archivos ndn_mgm y *ndn_mgmd* a la carpeta *"/usr/bin"* para que puedan ser ejecutados por el usuario.

6.2 CONFIGURACIÓN DE LOS NODOS DE ADMINISTRACIÓN

Para la iniciación de los servicios de MySQL Cluster se deben crear archivos de configuración para indicarle al programa cuáles son los nodos y qué función cumplen además de indicar en dónde se guardarán los datos. El primer archivo de configuración se llamará config.ini, este estará en *"/var/lib/mysql-cluster"* directorio que debe ser creado antes.

El archivo se llamará config.ini y deberá llevar dentro la siguiente información.

[NDBD DEFAULT] #Configuración por defecto del demonio de los nodos de datos
NoOfReplicas=2 #Número de replicas
DataMemory=80M #Capacidad de almacenamiento del clúster por default.
IndexMemory=18M#Capacidad de almacenamiento de índices.
[MYSQLD DEFAULT]
[NDB_MGMD DEFAULT] #Configuración por defecto del demonio del nodo de administración
DataDir=/var/lib/mysql-cluster #Directorio en el cual se almacenarán los datos de la base de datos
[TCP DEFAULT]
#Sección para las características de los nodos de administración.
[NDB_MGMD]
NodeId=1
#Dirección IP para el primer nodo de administración.
HostName=10.10.0.1
[NDB_MGMD]
NodeId=2
#Dirección IP para el segundo nodo de administración.
HostName=10.10.0.4
#Sección para las características de los nodos de datos.
[NDBD]
#Dirección IP para el primer nodo de almacenamiento.
HostName=10.10.0.2
DataDir= /var/lib/mysql-cluster
[NDBD]
#Dirección IP para el segundo nodo de almacenamiento.
HostName=10.10.0.3
DataDir= /var/lib/mysql-cluster
 # Una etiqueta [MYSQLD] para cada uno de los nodos SQL.
[MYSQLD]
[MYSQLD]

Nota: Es indispensable asignar un identificador a cada uno de los nodos de administración (NodeId) a diferencia de los demás nodos.

6.3 CONFIGURACIÓN DE LOS NODOS DE ALMACENAMIENTO

1. En los nodos de almacenamiento, se crea en cada nodo en el directorio /etc/ un archivo llamado “my.cnf”, donde irán los datos de conexión de cada una de dichas máquinas. Se debe hacer lo anterior en ambos nodos.

Imagen 1. Edición de archivo de configuración de MySQL "my.cnf".

2. Se crea un enlace simbólico llamado "*mysql*" desde el archivo "*mysql-cluster*" que se descargó anteriormente en cada una de las máquinas.
3. Se ubica el directorio con el enlace simbólico y se instalan actualizaciones del kernel del sistema operativo y además se instalan las bases de datos por defecto que tiene MySQL bajo el usuario "mysql" creado anteriormente y se le asigna un directorio para el almacenamiento de los datos.
4. Es necesario darle atributos de propietario al nuevo usuario "*mysql*" para poder iniciar los servicios.
5. Se mueven los archivos binarios que se encuentran en la carpeta de "*mysql*" que fue descargada a su nueva ubicación.

6.4 INICIO DE LOS SERVICIOS DE MYSQL CLUSTER

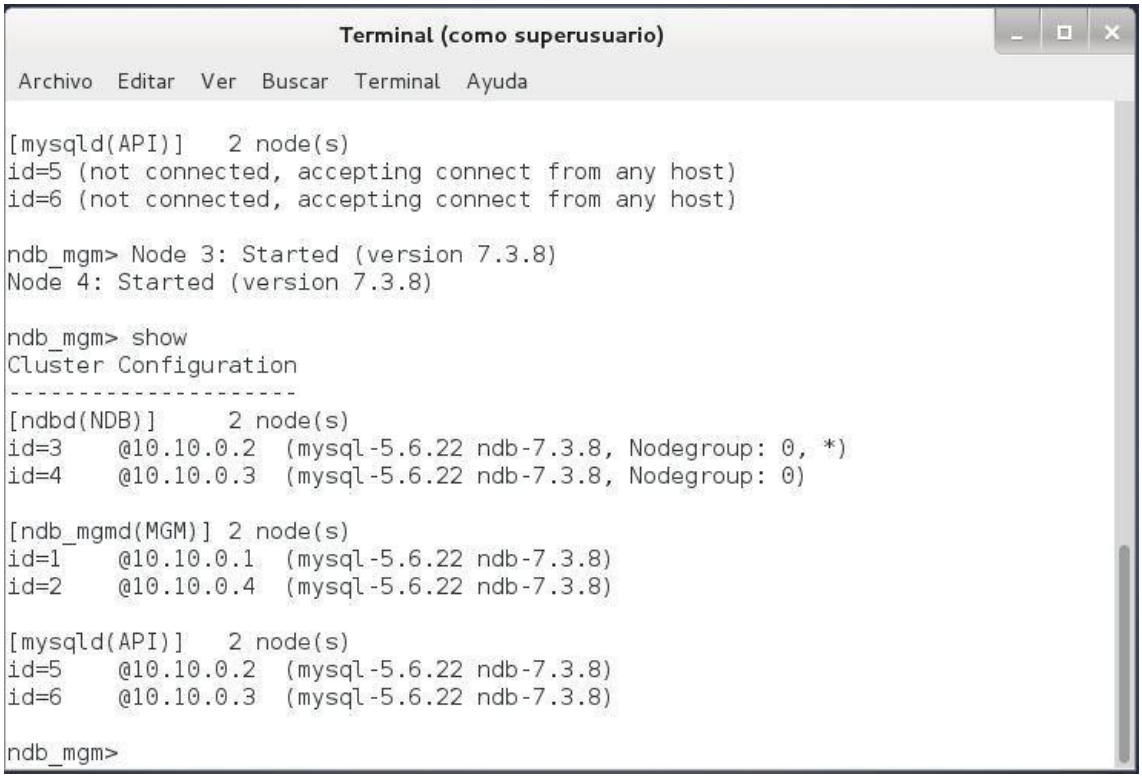
Para iniciar la operación del clúster es necesario iniciar los servicios de MySQL Cluster respectivos en cada uno de los nodos estableciendo un orden para evitar la generación de errores.

1. Se debe iniciar el servicio de los nodos de administración. Esto se debe realizar en los dos nodos de administración.
2. Se debe iniciar el servicio del demonio ndbd en cada una de las máquinas.
3. Iniciar el servicio de mysql server.

El demonio del nodo SQL es el que se encargará de hacer la replicación de los datos ingresados a la base de datos.

Desde el nodo administrador se puede ver que todos los nodos están conectados de manera correcta, para esto se usa el comando show como se muestra en la imagen.

Imagen 2. Verificación de inicio de los servicios.



Cuando los servicios ya pueden ser vistos desde la máquina 1 o desde la máquina 4 podemos crear, editar o eliminar bases de datos y ver cómo se replican los cambios, para esto se puede usar la terminal o algún Framework, en este caso se usó la terminal, desde la cual se hacen todas los cambios pertinentes como primera prueba de la replicación de los datos por medio del clúster.

6.5 INSTALACIÓN Y CONFIGURACIÓN DEL HAPROXY

En este proyecto se instaló HAProxy en los nodos administradores, para que balanceen la carga de las bases de datos. La estructura del balanceador es la siguiente:

Nodo 1 – Balanceador de carga (maquina1 y maquina4)
Hostname: haproxy
Private IP: 10.10.0.1 ó 10.10.0.4 (Según la IP del nodo administrador a configurar).

Nodo 2 – maquina2
Hostname: mysql-1
Private IP: 10.10.0.2 (IP del primer nodo SQL, en este caso "maquina2").

Nodo 3 – maquina3
Hostname: mysql-2
Private IP: 10.10.0.3 (IP del segundo nodo SQL, en este caso "maquina3").

A continuación se detallarán los pasos a seguir para la instalación y configuración del balanceador:

Primero se deben preparar los servidores MySQL mediante la creación de dos usuarios adicionales para HAProxy (haproxy_check y haproxy_root). El primer usuario será utilizado por HAProxy para comprobar el estado de un servidor.

Los campos que se deben configurar obligatoriamente para el correcto funcionamiento de HAProxy son User y Host (Nombre de usuario de HAProxy y la dirección IP correspondiente donde se va a alojar).

Se necesita también un usuario de MySQL con privilegios de root para acceder al clúster de MySQL desde HAProxy. El usuario root por defecto en todos los servidores se les permite ingresar sólo a nivel local. Si bien esto se puede solucionar mediante la concesión de privilegios adicionales para el usuario root, es mejor tener un usuario independiente con privilegios de root.

MySQL-Client tiene que estar instalado en el droplet de HAProxy para probar la conectividad, en caso de que no haya sido instalado previamente.

6.5.1 Instalación del haproxy

Para instalar HAProxy se necesita:

1. Habilitar el repositorio backports con la siguiente URL: *deb http://cdn.debian.net/debian wheezy-backports main*
2. Se debe habilitar que el HAProxy se inicie automáticamente, cuando se arranque la máquina.
3. Se ejecuta el script de inicio de HAProxy sin ningún parámetro. service haproxy

6.5.2 Configuración del haproxy

Cree y edite un nuevo archivo de configuración en el directorio de haproxy.

La primera parte de este archivo consta de la configuración por defecto de HAProxy

```
global
   log 127.0.0.1 local0 notice
   user haproxy
   group haproxy
#default SSL material locations
ca-base /etc/ssl/certs
crt-base /etc/ssl/private
#default ciphers to use on SSL-enabled listening sockets.

ssl-default-bind-ciphers kEECDH+aRSA+AES:+AES256:RC4-
SHA:!kEDH:!LOW:!EXP:!MD5:!aNULL:!eNULL

ssl-default-bind-options no-sslv3

defaults
   log global
   retries 2
   timeout connect 3000
   timeout server 5000
   timeout client 5000
```

La directiva log menciona un servidor "*syslog*" para que los mensajes de registro se envían.

Las directivas "*group"* y "*user*" cambian el proceso HAProxy al usuario / grupo especificado. Estos no se deben cambiar.

Los valores a ser modificados son las distintas directivas de tiempo de espera (*timeout*). La opción de conexión (*timeout connect*) especifica el tiempo máximo de espera para un intento de conexión a uno de los nodos SQL.

Los tiempos de espera de cliente y servidor (*timeout client*, *timeout server*) se aplican cuando se espera que el cliente o servidor para reconocer o enviar datos durante el proceso de TCP. HAProxy recomienda configurar el cliente y el servidor de tiempos de espera en el mismo valor.

La directiva "*retries"* define el número de reintentos para realizar en un Nodo SQL después de un fallo en la conexión.

4) A continuación, encontramos la parte principal del archivo de configuración.

```
listen mysql-cluster
   bind 0.0.0.0:3306
   mode tcp                          #protocolo de comunicación entre los nodos
   option mysql-check user haproxy_check          #usuario de haproxy
   balance roundrobin                    #Algoritmo que hace el balanceo
   server mysql-1 10.10.0.2:3306 check          #Nodo SQL (maquina2)
   server mysql-2 10.10.0.3:3306 check          #Nodo SQL (maquina3)
listen 0.0.0.0:8080
   mode http
   stats enable
```

stats uri /
stats realm Strictly\ Private
stats auth admin:admin1
stats auth user:user1

A diferencia del balanceador de carga HTTP, HAProxy no tiene un "*mode*" específico para MySQL así que por eso se utiliza "TCP". Hemos establecido HAProxy para escuchar 0.0.0.0 porque la aplicación (MySQL) no se encuentra alojada en ninguno de los nodos administradores; en caso de que lo esté, es recomendable hacerlo escuchar 127.0.0.1.

Una vez que haya terminado la configuración, inicie el servicio HAProxy.

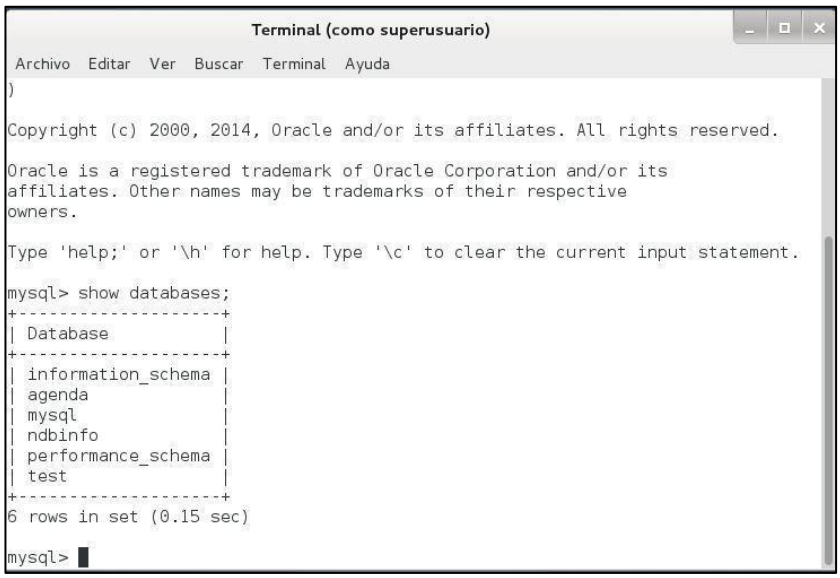
6.6 PRUEBAS Y VERIFICACIÓN

6.6.1 Prueba local

Para comenzar se debe ingresar a la api de *mysql* desde la cual se realizarán todas las operaciones pertinentes de la base de datos. Para ingresar a dicha api se debe ingresar en la terminal el comando *mysql –u root –p* donde –u hace referencia al usuario que en este caso es root y –p a la contraseña la cual para la prueba fue obviada.

MySQL por defecto tiene varias bases de datos creadas como se puede observar en la imagen. Para la consecución de la prueba fue necesario crear una base de datos con el nombre de ‘’agenda’’.

Imagen 3. Listado de bases de datos en el clúster.



Para esta prueba se usó la base de datos “agenda”, utilizando el siguiente comando: *use agenda*;

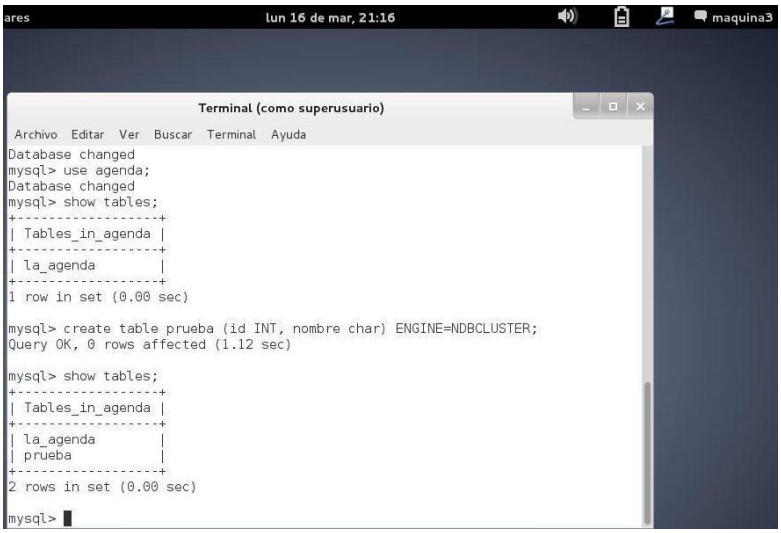
Para ver las tablas que componen la base de datos, se utilizó *show tables*, tal como se muestra en la imagen.

Imagen 4. Listado de tablas de la base de datos.

Luego de esto, se procede a crear una nueva tabla llamada “prueba” en la base de datos “agenda”. Además del comando usado para crear la tabla (CREATE TABLE) con sus respectivos campos, se debe asignar a la propiedad “ENGINE” de la tabla, el valor NDBCLUSTER.

Nota: Es de suma importancia que el valor de la propiedad “ENGINE” sea NDBCLUSTER, debido a que si utiliza otro valor como por ejemplo INNODB, no habrá replicación de los datos entre los nodos del clúster.

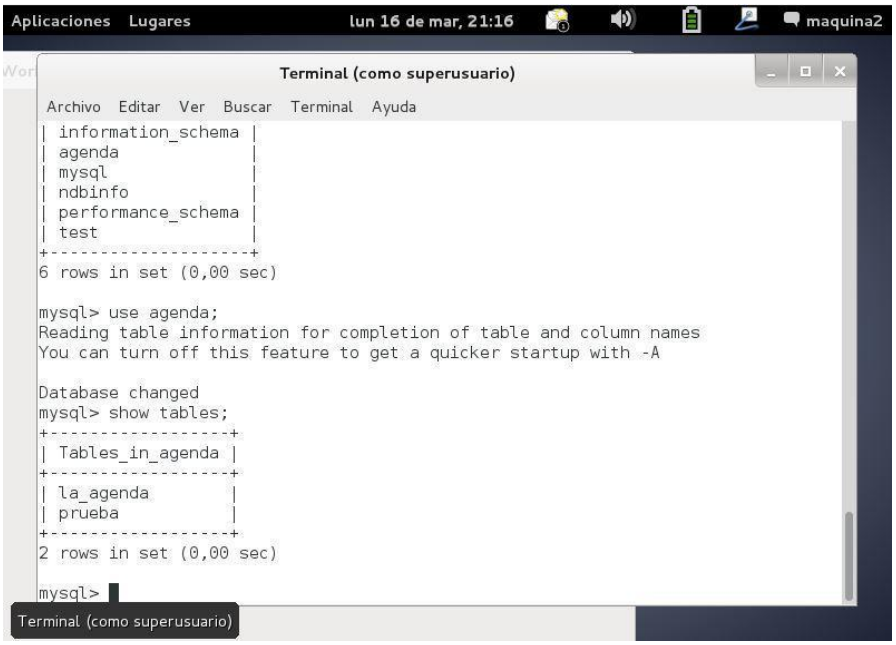
Imagen 5. Inserción de nueva tabla.



Después de ejecutar el comando *show tables*, se observa que la tabla ha sido creada correctamente en la máquina 3.

Ahora se puede observar como las tablas de la base de datos creadas desde la máquina 3 es replicadas automáticamente en la máquina 2, para esto se ejecuta el comando "*show tables;"*. La tabla puede ser accedida desde la máquina 2.

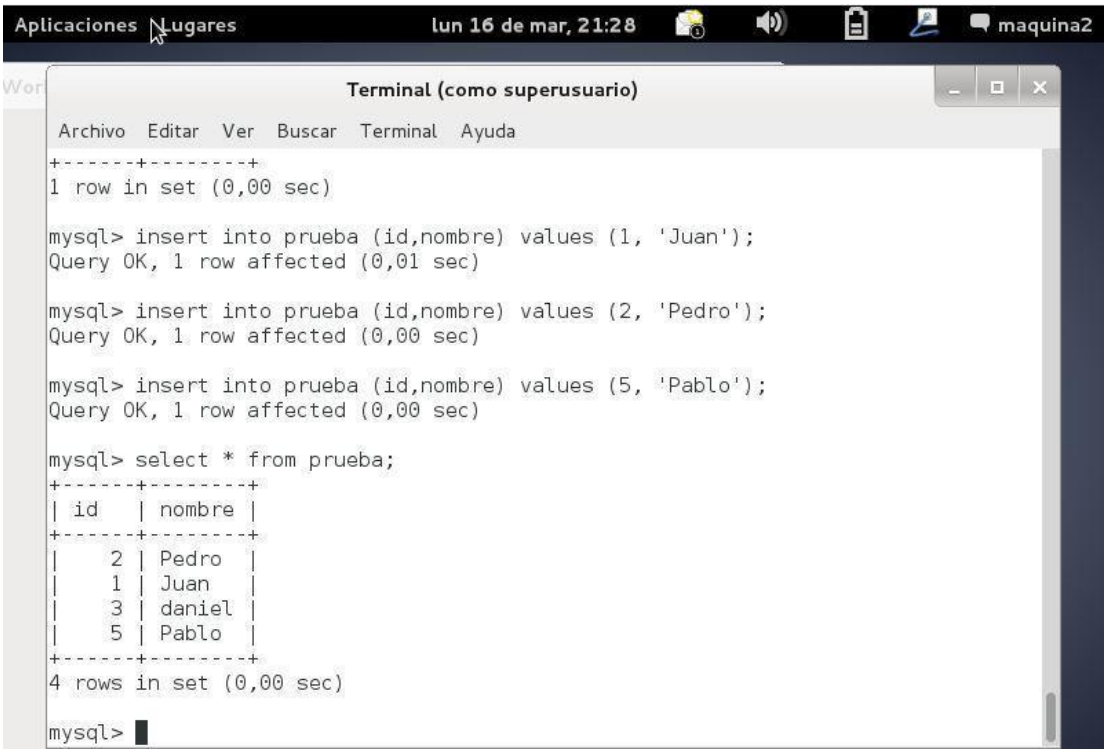
Imagen 6. Listado nuevo de tablas.



Inserción de datos:

Como ahora se puede acceder a la base de datos y sus tablas desde cualquiera de los nodos, se hacen inserción de datos desde la máquina 2 para poder ver su replicación en la "máquina3". En la siguiente imagen se puede observar la inserción de tres registros a la base de datos mediante el comando *insert into prueba (id,nombre) values (el id a registrar,'nombre a registrar').* Para verificar que los datos fueron insertados de manera correcta, se ingresa el comando *select * from prueba;* los datos ingresados se pueden apreciar correctamente en la imagen.

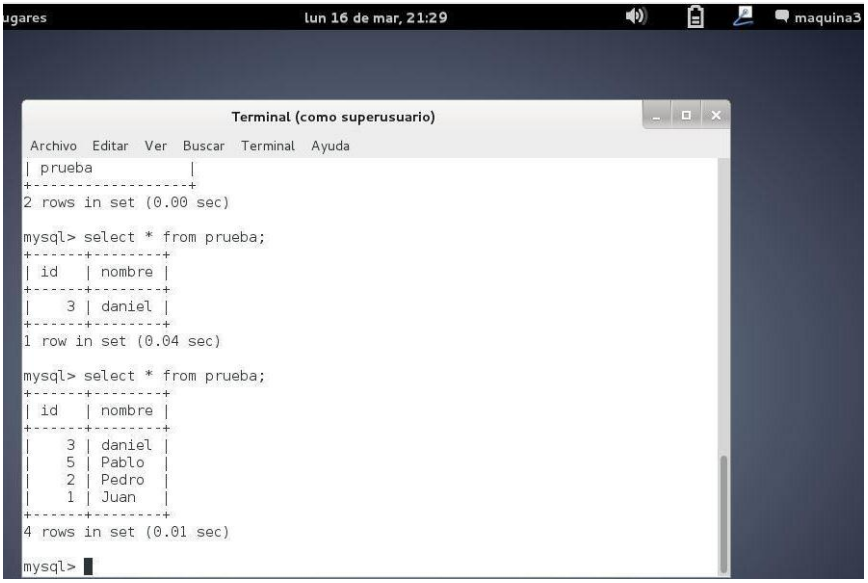
Imagen 7. Inserción de datos.



Automáticamente en "maquina3" quedan guardados los nuevos registros, insertados en "maquina2", con lo cual se observa que la replicación es inmediata y sin inconsistencias.

Mediante el comando *select * from prueba;* efectivamente se obtienen los datos insertados desde "maquina2", tal y como lo muestra la imagen.

Imagen 8. Listado de datos insertados.

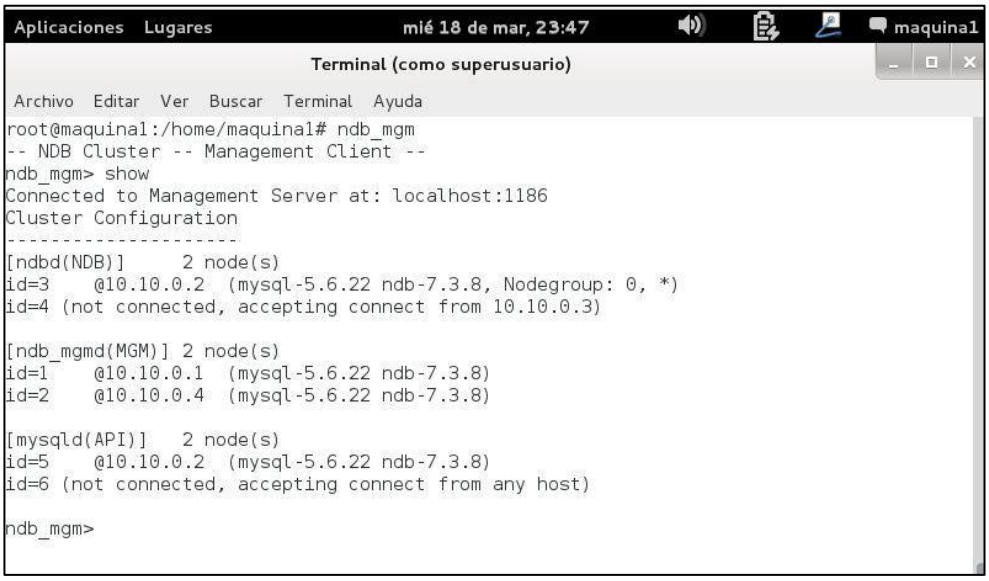


Eliminación e inserción de datos con máquinas inactivas

Con lo anteriormente mostrado se ha probado entre otras cosas la replicación e integridad de los datos, a continuación se probará la mantenibilidad de las transacciones, eliminando un registro desde "maquina2" pero con la "maquina3" apagada.

Para esto, se procede a apagar "maquina3" y se corrobora yendo al nodo administrador (maquina1) y dentro del cliente administrador del clúster se ejecuta el comando *show*, tal y como se muestra en la siguiente imagen.

Imagen 9. Caída de la "maquina3".



Después corroborar que la "máquina3" por medio del nodo administrador, se hace la eliminación de uno de los registros anteriormente ingresados desde la "máquina2" para que al encender de nuevo la "máquina3" se pueda verificar si el registro se replica automáticamente. Para hacer la eliminación del campo se usa el comando "*delete from prueba where nombre='campo a eliminar'"*. Se usa el comando "*select * from prueba"* para listar los registros existentes luego de la eliminación.

Imagen 10. Eliminación de registros.

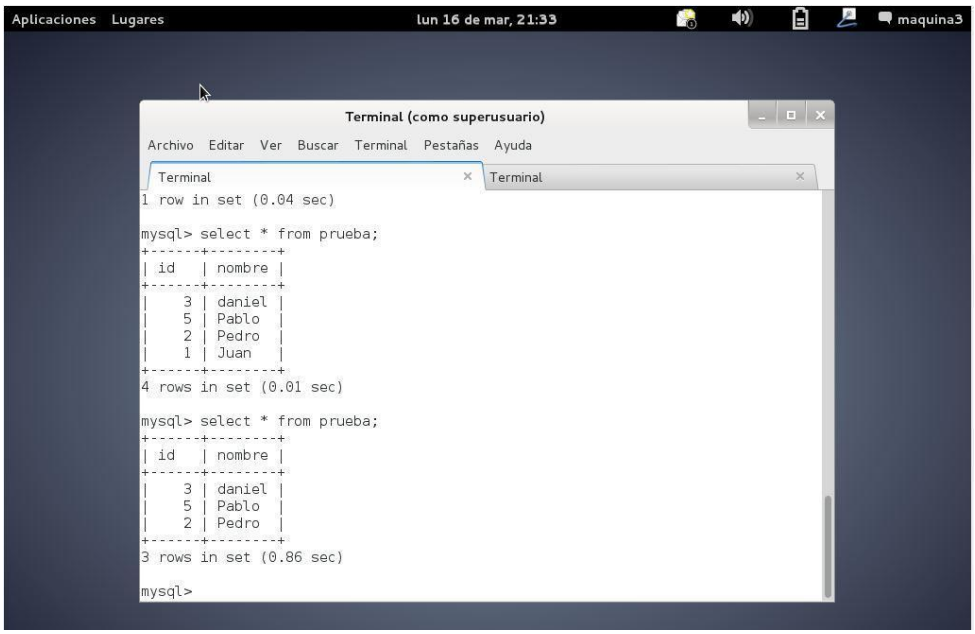
Luego de ver que se ha eliminado un registro en "maquina2", se arranca el nodo de datos que estaba apagado (maquina3) y se conecta nuevamente dicha máquina al clúster. Al refrescar la terminal del nodo administrador o al volver a ejecutar el comando *show*, se observa que están conectados todos los nodos nuevamente.

Imagen 11. Levantamiento de la "maquina3".

```
ndb_mgm> show
Cluster Configuration
---------------------
[ndbd(NDB)]     2 node(s)
id=3    @10.10.0.2  (mysql-5.6.22 ndb-7.3.8, Nodegroup: 0, *)
id=4    @10.10.0.3  (mysql-5.6.22 ndb-7.3.8, Nodegroup: 0)

[ndb_mgmd(MGM)] 2 node(s)
id=1    @10.10.0.1  (mysql-5.6.22 ndb-7.3.8)
id=2    @10.10.0.4  (mysql-5.6.22 ndb-7.3.8)

[mysqld(API)]   2 node(s)
id=5    @10.10.0.2  (mysql-5.6.22 ndb-7.3.8)
id=6    @10.10.0.3  (mysql-5.6.22 ndb-7.3.8)

ndb_mgm>
```

Para finalizar la prueba, desde la "máquina3", se debe ingresar de nuevo a la api de MySQL y seleccionar la base de datos "agenda". Se usa el comando "*select * from prueba"* para afirmar que la eliminación desde la "máquina2" es replicada al iniciar de nuevo los servicios.

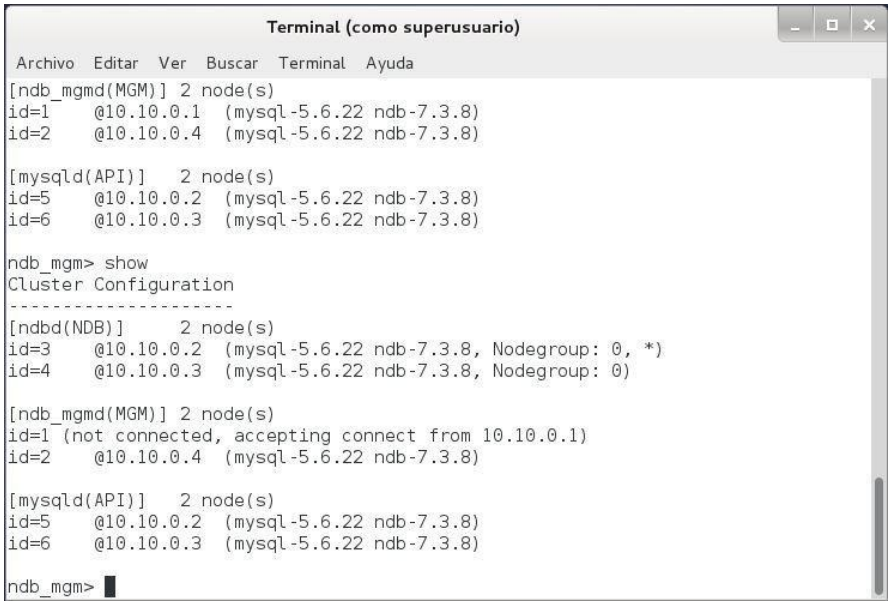
Imagen 12. Verificación de eliminación.



Finalmente, se ha probado la mantenibilidad de las transacciones, lo cual también asegura la integridad de los datos, mayor confiabilidad y alta disponibilidad del sistema.

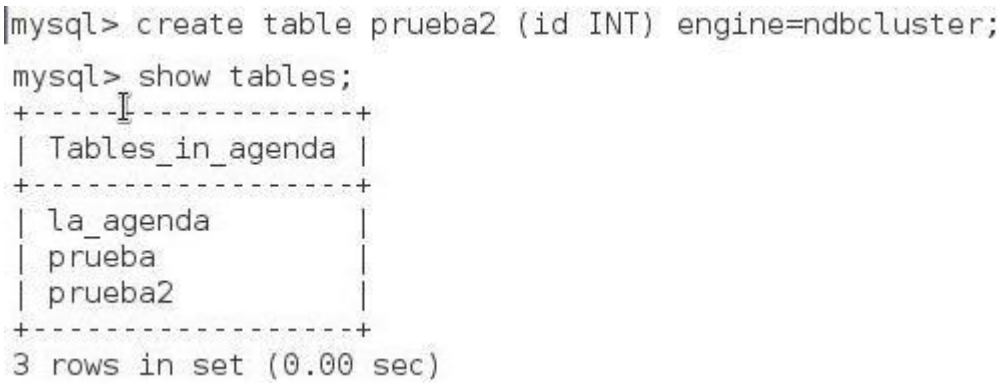
Esta parte de la prueba se realiza con el fin de verificar si cuando uno de los nodos de administración falla los datos se siguen replicando. El nodo que se apaga es el número uno como se puede observar en la imagen.

Imagen 13. Caída del nodo administrador "maquina1".



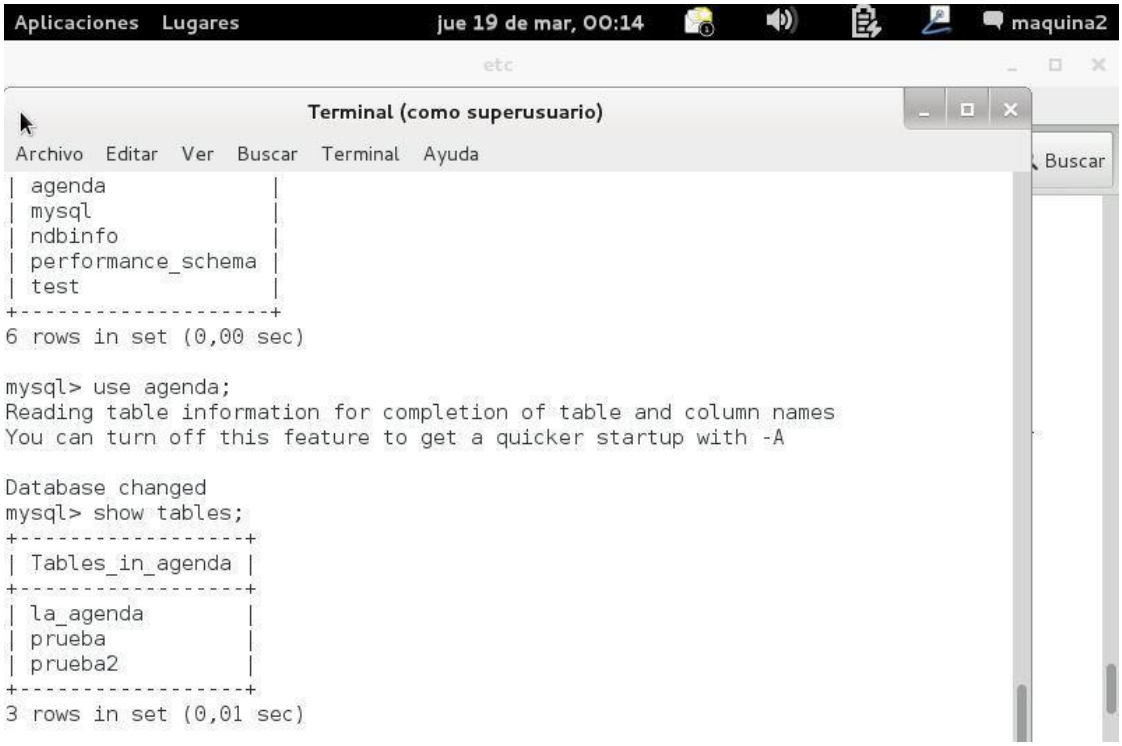
Luego de apagar el nodo 1 se crea una nueva tabla desde la máquina 3 para después ver la replicación desde la máquina 2.

Imagen 14. Creación de una tabla sin nodo administrador "maquina1".



Para comprobar la replicación se usa el comando *"show tables"* desde la máquina 2 y se observa que se hizo la replicación automáticamente lo cual se concluye que el clúster sigue funcionando de manera óptima con sólo uno de los nodos de administración.

Imagen 15. Comprobación de creación de la nueva tabla.



La siguiente prueba consiste en manipular la base de datos desde una aplicación web que simulará en pequeña escala el trabajo que haría el clúster en el aula virtual MEIWEB. Esta aplicación fue suministrada por el grupo de investigación GID-CONUSS.

6.6.2 Prueba con aplicativo web

La siguiente prueba consiste en manipular la base de datos desde una aplicación web que simulará en pequeña escala el trabajo que haría el clúster en el aula virtual MEIWEB. Esta aplicación fue suministrada por el grupo de investigación GID-CONUSS y adaptada a las necesidades de los autores.

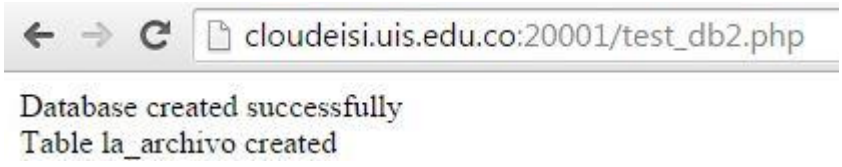
Para comenzar se aclara que la base de datos a usar es la misma que se usó para la prueba anterior.

Lo primero que se hace es acceder a la aplicación web que fue previamente alojada en el servidor del grupo de investigación GID-CONUSS, esta aplicación simula una agenda, en la cual se pueden agregar el nombre, correo, teléfono fijo y teléfono móvil de un contacto, además desde ella misma se realiza la creación de una nueva tabla llamada “la_archivo” con el objeto de subir archivos al servidor y almacenar la ruta en donde quedan guardados dichos archivos.

Imagen 16. Interfaz de la aplicación de prueba.

Para iniciar se debe crear la base de datos "agenda" e insertar la tabla "la_agenda" dando clic en el primer enlace "Crear base de datos Agenda con inserción".

Imagen 17. Creación de la base de datos.



Inserción de datos

Lo siguiente a realizar con la aplicación es insertar datos. Las primeras inserciones se realizan con el clúster en total funcionamiento, luego se realizan pruebas de inserción de archivos en donde se corrobora, por medio de la caída de alguno de los servicios del clúster, que hay alta disponibilidad en la base de datos.

Los datos son insertados a través del enlace de "Añadir un registro" así como se puede ver en las siguientes imágenes.

Imagen 18. Inserción de registros.

Imagen 19. Inserción satisfactoria.

Por medio del enlace "Listado completo" se visualizan los registros insertados en la base de datos anteriormente.

Imagen 20. Total de registros insertados.

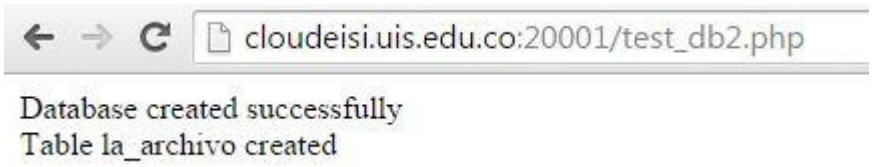
Para continuar la prueba es necesario tener abajo el nodo administrador "maquina4" y realizar la creación de la tabla "la_archivo" para así subir archivos al sistema, todo esto se evidencia en las próximas imágenes.

Como se puede observar, se corrobora la caída del nodo administrador "maquina4" para luego crear por medio de la aplicación la tabla "la_archivo", en la cual se inserta el nombre del archivo, la ruta del sistema en que se guardan los archivos de usuario y el tipo de archivo agregado.

Imagen 21. Caída nodo administrador "maquina4".

```
root@maquina1:~# ndb_mgm
-- NDB Cluster -- Management Client --
ndb_mgm> show
Connected to Management Server at: localhost:1186
Cluster Configuration
---------------------
[ndbd(NDB)]     2 node(s)
id=3    @10.10.80.2  (mysql-5.6.22 ndb-7.3.8, Nodegroup: 0)
id=4    @10.10.80.3  (mysql-5.6.22 ndb-7.3.8, Nodegroup: 0, *)

[ndb_mgmd(MGM)] 2 node(s)
id=1    @10.10.80.1  (mysql-5.6.22 ndb-7.3.8)
id=2 (not connected, accepting connect from 10.10.80.4)

[mysqld(API)]   2 node(s)
id=5    @10.10.80.3  (mysql-5.6.22 ndb-7.3.8)
id=6    @10.10.80.2  (mysql-5.6.22 ndb-7.3.8)
```

Imagen 22. Inserción tabla "la_archivo".



Se puede ver, por medio de la aplicación cómo se selecciona el archivo necesitado por el usuario y la manera de enviarlo al servidor para ser guardado.

Imagen 23. Selección del archivo a subir.

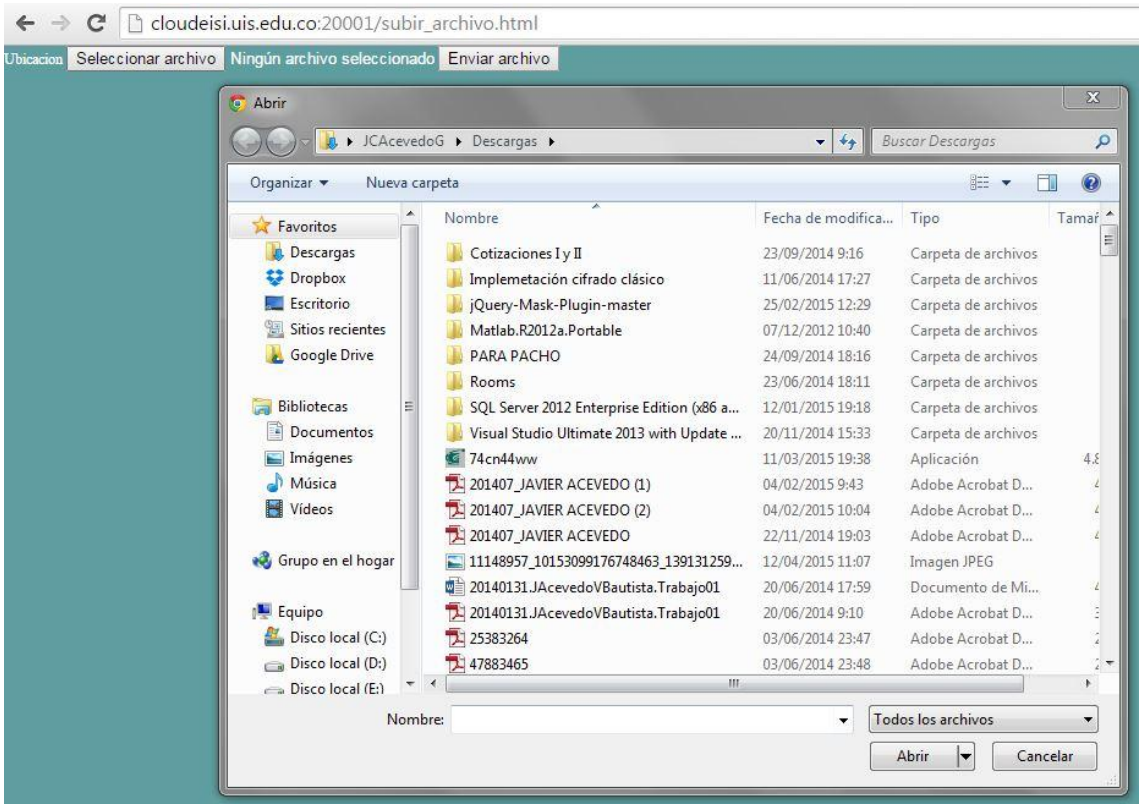
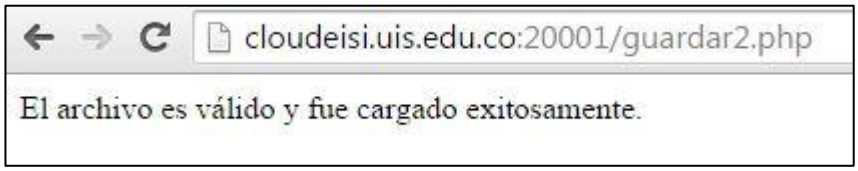


Imagen 24. Archivo cargado.

Imagen 25. Carga de archivo satisfactoria.



Como lo fundamental en todas las pruebas es comprobar que el clúster realice de manera óptima la replicación del clúster, en necesario dirigirse a cualquiera de los nodos de datos (“maquina2” y “maquina3”) y ver, por medio de una consulta “*select*”, la ya mencionada replicación en los nodos.

Es necesario recalcar que la caída de cualquiera de los nodos de administración no afecta en lo absoluto al buen comportamiento del clúster.

Imagen 26. Verificación de archivos subidos.

```
mysql> select * from la_archivo;
+----+--------------+-----------------------------------------+-----------------+
| id | nombre       | contenido                               | tipo            |
+----+--------------+-----------------------------------------+-----------------+
|  2 | 25383264.pdf | /var/www/AgendaBD/Archivos/25383264.pdf | application/pdf |
+----+--------------+-----------------------------------------+-----------------+
1 row in set (0.00 sec)
```

Inserción de datos con máquinas inactivas

Para verificar el funcionamiento del sistema de bases de datos distribuidas de manera más rigurosa, se hace indispensable hacer una inserción más, pero esta vez con un fallo en alguno de los nodos de datos. Para el caso de estudio el nodo es el "maquina3" para luego consultar desde el nodo de datos restante si el archivo es subido al servidor.

Imagen 27. Caída nodo "maquina3"

```
ndb_mgm> Node 4: Node shutdown completed. Initiated by signal 15.

ndb_mgm> show
Cluster Configuration
---------------------
[ndbd(NDB)]     2 node(s)
id=3    @10.10.80.2  (mysql-5.6.22 ndb-7.3.8, Nodegroup: 0, *)
id=4 (not connected, accepting connect from 10.10.80.3)

[ndb_mgmd(MGM)] 2 node(s)
id=1    @10.10.80.1  (mysql-5.6.22 ndb-7.3.8)
id=2 (not connected, accepting connect from 10.10.80.4)

[mysqld(API)]   2 node(s)
id=5 (not connected, accepting connect from any host)
id=6    @10.10.80.2  (mysql-5.6.22 ndb-7.3.8)
```

Imagen 28. Archivo subido con caída del nodo "maquina3".

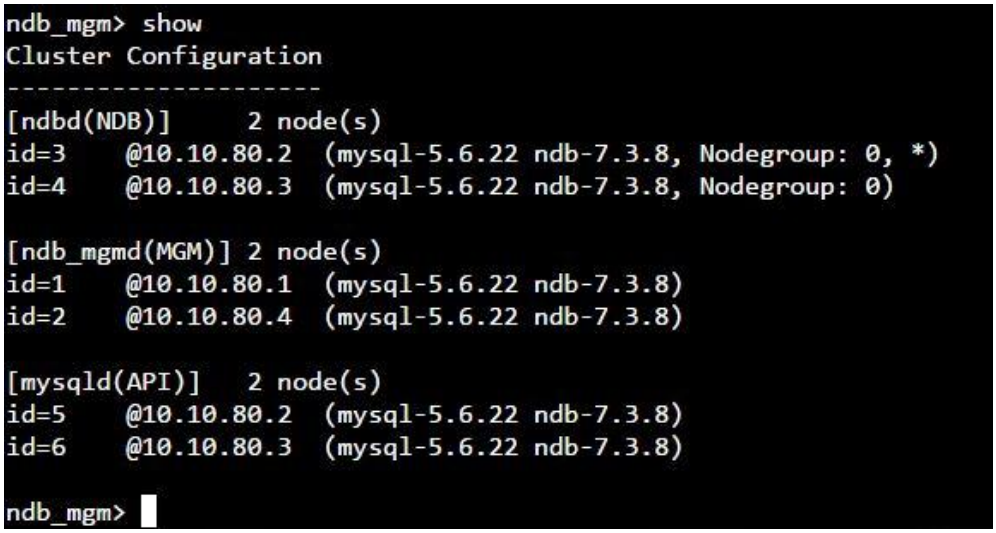
```
mysql> select * from la_archivo;
+----+--------------+-----------------------------------------+-----------------+
| id | nombre       | contenido                               | tipo            |
+----+--------------+-----------------------------------------+-----------------+
|  3 | Escaneo10.jpg | /var/www/AgendaBD/Archivos/Escaneo10.jpg | image/jpeg      |
|  2 | 25383264.pdf  | /var/www/AgendaBD/Archivos/25383264.pdf  | application/pdf |
+----+--------------+-----------------------------------------+-----------------+
2 rows in set (0.00 sec)
```

Verificación de replicación

Para continuar la prueba se vuelven a levantar los servicios caídos y se verifica que la replicación se haga de forma satisfactoria.

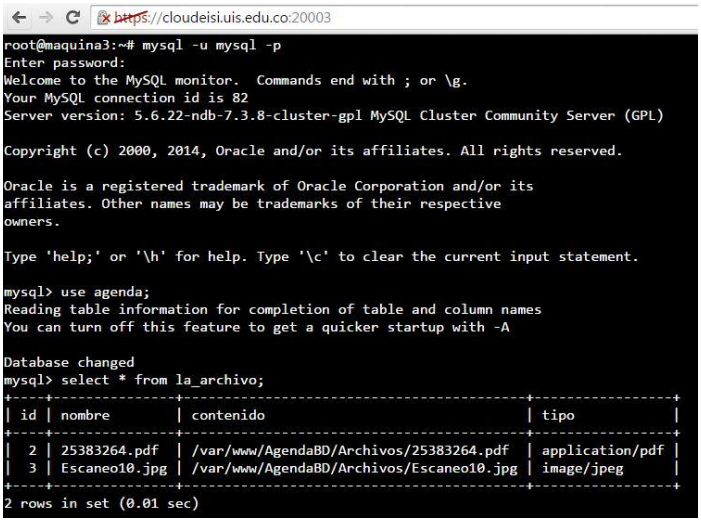
En este caso, se puede observar la configuración del clúster con todos los servicios iniciados y funcionando.

Imagen 29. Clúster con todos los servicios funcionando.



Luego, desde el nodo de datos "maquina3" se observa que la replicación de los datos insertados en ausencia de su servicio es realizada tan pronto ésta máquina vuelve a estar arriba y con todos los servicios funcionando.

Imagen 30. Verificación de replicación desde el nodo "maquina3".



Eliminación de datos

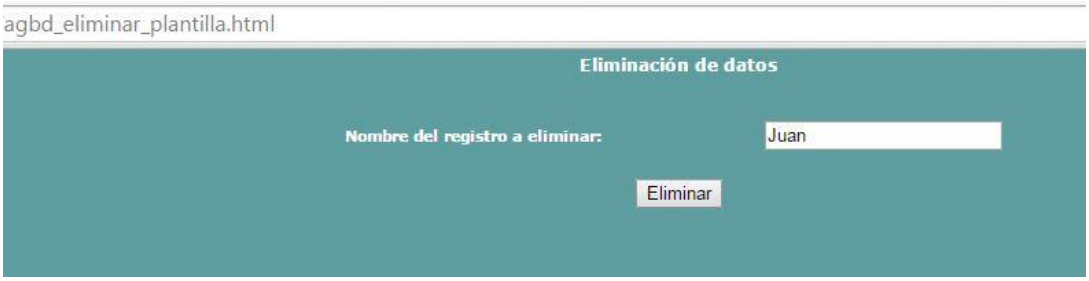
Para finalizar la prueba, se busca mirar la replicación de una transacción de eliminación de registros cuando alguno de los nodos de datos está caído, en este caso el nodo caído será el "maquina2" como se evidencia a continuación.

Imagen 31. Servicios del nodo "maquina2" no iniciados.

```
ndb_mgm> show
Cluster Configuration
---------------------
[ndbd(NDB)]     2 node(s)
id=3 (not connected, accepting connect from 10.10.80.2)
id=4    @10.10.80.3  (mysql-5.6.22 ndb-7.3.8, Nodegroup: 0, *)

[ndb_mgmd(MGM)] 2 node(s)
id=1    @10.10.80.1  (mysql-5.6.22 ndb-7.3.8)
id=2    @10.10.80.4  (mysql-5.6.22 ndb-7.3.8)

[mysqld(API)]   2 node(s)
id=5 (not connected, accepting connect from any host)
id=6    @10.10.80.3  (mysql-5.6.22 ndb-7.3.8)

ndb_mgm>
```

Basándose en los registros anteriormente insertados se hace la eliminación sugerida para analizar la replicación de los datos cuando los servicios caídos por causa del fallo vuelven a establecerse.

Imagen 32. Eliminación de un registro.



Después de esta eliminación se vuelven a iniciar los servicios en el nodo "maquina2" y por medio de un "*select"* se puede observar que la replicación se realiza con éxito.

Imagen 33. Verificación de eliminación desde el nodo "maquina2".

```
mysql> select * from la_agenda;
+--------+-----------------+------------+------------+
| nombre | correoe         | tlf_fijo   | tlf_movil  |
+--------+-----------------+------------+------------+
| Nohora | nohora@gmail.com | 3118284689 | 3118284689 |
| Daniel | daniel@gmail.com | 3123458765 | 3123458765 |
+--------+-----------------+------------+------------+
2 rows in set (0.00 sec)
```

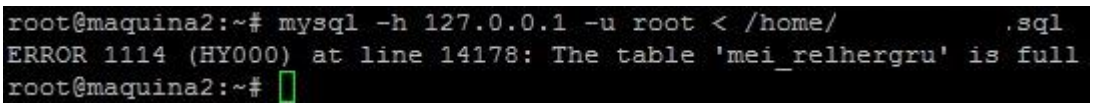
6.6.3 Prueba con meiweb

En las pruebas anteriores se pudo comprobar que la función principal de MySQL Cluster, la cual es replicar, se cumple a cabalidad aun cuando alguno de los nodos tiene una falla o cuando hay algún tipo de transacción CRUD.

A continuación lo primero que se realizó fue la creación de un Back up de la base de datos de producción, luego en el archivo generado con nombre Basededatos.sql se realizó el cambio del motor de almacenamiento de las tablas a ndbcluster. Después de este proceso, se creó la base de datos en MySQL Cluster con el comando

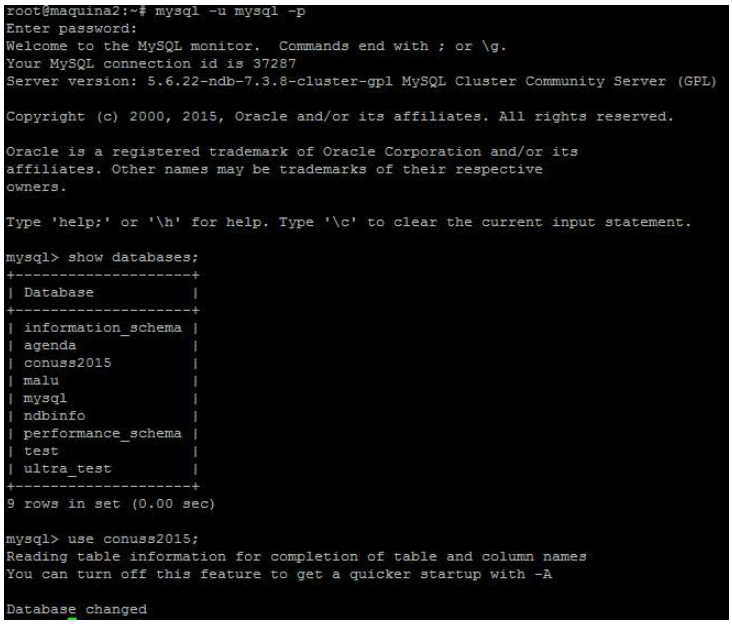
"mysql –h 127.0.0.1 –u root </home/Basededatos.sql"

Imagen 34. Importación de la base de datos.



Como se puede observar en la anterior imagen, después de ejecutar el comando para cargar la base de datos, se genera un error, específicamente el error "1114" que hace referencia a que al crear la base de datos y sus respectivas tablas con sus registros se llega a un máximo datos almacenados y no permite seguir haciendo la creación de las tablas restantes. El total de tablas en la base de datos es de 95 y sólo son importadas 60.

Imagen 35. Selección de la base de datos insertada.



En el archivo de configuración (config.ini) de MySQL Cluster existen dos ítems llamados DataMemory e IndexMemory, que sirven para determinar la capacidad de almacenamiento de datos. Tan pronto se generó el error, el DataMemory se encontraba por default en 80 Megabytes y el IndexMemory en 18 Megabytes pero la creación de la base de datos no fue satisfactoria, a pesar de que el tamaño de la base de datos es de menos cuantía. Conforme a lo anterior se decide aumentar el valor de dichos ítems a 256 Megabytes cada uno, pero tampoco es satisfactorio el proceso. Sucesivamente se les aumenta el tamaño hasta llegar a 1024 Megabytes (1 Gigabyte) pero tampoco es posible terminar de importar la base de datos.

El resultado de lo anterior lleva a consultar más a fondo el origen del error, en este proceso se encontraron varias hipótesis, que fueron descartadas una por una con cambios realizados en la configuración del clúster. Como primera instancia se encontró que la cantidad de memoria RAM usada en cada máquina podría influir, por eso se decidió aumentar la RAM de todo el clúster, de 1 Gigabyte a 2, pero esto no genero ningún cambio en la importación de la base de datos, ni siquiera se logró insertar un registro más en la misma base de datos, o desde cualquier otra alojada dentro del clúster, por lo que se determinó que la RAM no sería el factor del error. También se encontró que la arquitectura de las máquinas usadas en el clúster supondría ser el problema, por eso se cambió la arquitectura de la máquinas virtuales de 32 Bites a 64 Bites y después hacer la configuración del clúster desde cero, pero ésta tampoco fue la respuesta al error. [Ver Anexo A.]

Después de la prueba anterior, se probó fragmentar la base de datos en dos partes, un fragmento de 45 tablas y el otro de 40. Cuando se hace la inserción del primer fragmento, todas las tablas y su contenido son agregados de manera correcta, pero cuando se hace la inserción del segundo fragmento, sólo se logran agregar de 15 tablas de las 40 restantes llegando así al error que se había presentado anteriormente. Para poder fragmentar la base de datos se tuvo que consultar acerca de los tipos de fragmentación y sus ventajas, en donde se halló que es posible hacer fragmentos de las tablas, lo cual implicaba rediseñar la base de datos del MEIWEB y además ingresar a la codificación de la aplicación, a la cual por cuestiones de seguridad no se tiene acceso.

Al no encontrar una respuesta clara del suceso del error se decide ir más a fondo en la consulta del mismo y se evidencia lo siguiente:

Para empezar, se encuentra que a diferencia de los motores de almacenamiento Innodb y Myisam, el engine ndbcluster no reconoce diferencias entre los tipos de dato varchar y char.

Como es sabido un tipo de dato varchar, a diferencia de un char sólo usa el espacio de datos usados en la inserción independientemente del valor que se le haya establecido, pero como se mencionó, ndbcluster lo asimila como un tipo char.

Además de esto también se encuentra algo determinante en las especificaciones de datos del MySQL Cluster que dice que el máximo número de objetos con un motor ndbcluster -en los cuales se contemplan bases de datos, tablas e índices- es limitado a un volumen de 20320, por lo que no es posible hacer la importación de la base de datos completa, ya que ésta contiene un gran volumen de objetos. La cantidad de objetos de la base de datos del aula virtual MEIWEB es aproximadamente de 12000, pero al implementarla en MySQL Cluster esta cantidad se duplica a causa del método de replicación superando así, el límite permitido por la herramienta. [Ver Anexo B.]

Lo anterior es una limitante de la versión actual de MySQL Cluster, que está siendo estudiada para su mejoramiento en versiones futuras. Es importante aclarar que a pesar que la base de datos no pudo ser importada en su totalidad el funcionamiento del clúster en cuanto a replicación se refiere no se ve afectado.

7. CONCLUSIONES

Se incursionó en el estudio de los sistemas de bases de datos distribuidas, especialmente en las bases de datos distribuidas de MySQL, en donde se encontró que dichos sistemas aún no son usados de manera masiva en la academia a diferencia de la industrial, en donde, por ejemplo, los bancos tiene predilección por bases de datos distribuidas con fragmentación, dado que estos sistemas se amoldan fácilmente a la estructura de dependencias que ellos tienen.

Se realizó un prototipo de bases de datos distribuido con MySQL Cluster, que se implementó en el servidor del grupo de investigación GID-CONUSS, dicho prototipo fue sometido a distintas pruebas, entre esas se encuentran las de funcionalidad, integridad de datos, mantenibilidad de transacciones y tolerancia a fallos, lo que genera una mayor confiabilidad y alta disponibilidad de la base de datos.

Debido al diseño que se encontró en el sistema de bases de datos del aula virtual MEIWEB, la cual está desarrollada en MySQL, y con base al estudio del estado del arte, se determinó gracias a la compatibilidad de las distribuciones de MySQL y MySQL Cluster, que no fue necesario hacer un nuevo diseño de la base de datos, ya que MySQL Cluster soporta bases de datos diseñadas para MySQL básico, sólo necesitando que se cambie el motor de almacenamiento o Engine al usado por MySQL el cual se llama “ndcluster”; este último es el que permite hacer la replicación de todas las transacciones realizadas en la base de datos.

La implementación de este nuevo sistema de bases de datos supone un nuevo desafío para el grupo de investigación GID-CONUSS ya que el sistema ya mencionado requiere personal con mejores capacidades para el mantenimiento y la administración del sitio, además de tener visión de futuras mejoras.

Los sistemas de bases distribuidas MySQL Cluster requieren de más memoria RAM para su funcionamiento que una base centralizada de MySQL, dado que deben realizar la réplica de las transacciones de manera automática para evitar incongruencias.

8. RECOMENDACIONES

Montar máquinas virtuales con sistemas operativos de arquitectura de 64-bits, para aumentar el rendimiento del clúster.

Asignar la mayor cantidad de espacio en memoria (RAM) posible a las máquinas virtuales para evitar problemas al cargar datos en el clúster.

Utilizar imagen del sistema operativo usado (en este caso Debian) sin interfaz gráfica, para optimizar la RAM de las máquinas virtuales.

Basados en los resultados obtenidos en este proyecto, no es recomendable alojar bases de datos relativamente grandes en el clúster, debido a las limitaciones que presenta MySQL Cluster.

En el momento de crear los usuarios de mysql que accederán a las bases de datos requeridas, es de suma importancia otorgarles todos los permisos y privilegios a dichos usuarios, con el fin de que no se presenten inconvenientes en el manejo de las bases de datos.

Una de las funcionalidades más importantes de un sistema de bases de datos distribuidas, es que debe funcionar independientemente de la red a la cuál esté conectado, por eso es indispensable configurar una red interna para que se comuniquen entre sí los nodos que hacen parte del clúster.

Investigar la manera de mejorar el funcionamiento del clúster armado en el presente proyecto, con el objetivo de implementar el sistema de bases de datos distribuidas en el aula virtual MEIWEB.

En caso de necesitar otro nodo en el clúster, se recomienda hacer una copia de las bases de datos que haya, porque es necesario reiniciar completamente el servicio y cambiar la configuración para agregar el nuevo nodo, de esta manera se evita que haya pérdida de información.

Se recomienda para futuras investigaciones en este tema, consultar otras herramientas utilizadas para la configuración y administración de bases de datos distribuidas. Actualmente además de la herramienta utilizada en este proyecto se encuentran otras libres, tales como: Galera Cluster, Percona XtraDB Cluster, MariaDB Galera Cluster.

BIBLIOGRAFÍA

- ABREUS Yisel Alfaro, "Tipos de fragmentación en bases de datos distribuidas", Empresa de telecomunicaciones de Cuba SA, Ciudad de la Habana, Cuba, Diciembre de 2015.

- AGUILERA DÍAZ Rubén. Montar un cluster de MySQL. {En línea}. {10 de Enero de 2015} Disponible en: (http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=MySQLCluster)

- ATTARD James, MySQL with HAProxy for Failover and Load Balancing. {En línea} {17 de enero de 2015}. Disponible en: (http://www.jamesattard.com/2014/09/mysql-with-haproxy-for-failover-and.html)

- Capítulo 16. MySQL Cluster {En línea}. {13 de febrero de 2015} Disponible en: (http://manuales.guebs.com/mysql-5.0/ndbcluster.html#multi-config)

- COSTILLA Carmen, "Arquitecturas de los SGBD distribuidos", Departamento de Ingeniería de Sistemas Telemáticos, Universidad Politécnica de Madrid, Madrid, Diciembre de 2009.

- GUTIÉRREZ COSÍO Celia. DISEÑO DE UN SISTEMA GESTOR DE BASE DE DATOS DISTRIBUIDA BASADO EN ORACLE9i, Revista "Tecnología y desarrollo", Universidad Alfonso X El Sabio, Madrid 2008.

- How to setup mysql-cluster (beginners tutorial). {En línea}. {17 de Enero de 2015}. Disponible en: (https://cyrenity.wordpress.com/2010/08/12/howto-setup-mysql-cluster-beginners-tutorial/)

- Instalación, Configuración y demostración de un MySQL Cluster. {En línea}. {10 de Diciembre de 2014} Disponible en: (http://informacion-activa.blogspot.com/2012/04/instalacion-configuracion-y.html)

- Instalación de MySQL Cluster (ndb). {En línea}. {15 de Enero de 2015} Disponible en: (http://systemadmin.es/2012/01/instalacion-de-mysql-cluster-ndb)

- JESIN A. How To Use HAProxy to Set Up MySQL Load Balancing. {En línea}. {2 de Marzo de 2015}. Disponible en: (https://www.digitalocean.com/community/tutorials/how-to-use-haproxy-to-set-up-mysql-load-balancing--3)

- MEJÍA QUINTANILLA Verónica Esmeralda, JIMÉNEZ MARTÍNEZ Nelson Arnoldo, CANIZÁLEZ RAMÍREZ Mauricio Armando, ALFARO AVILA William Ernesto. Bases de datos distribuidas con MySQL. {En línea}. { 22 de Enero de 2015} Disponible en: (http://basesdedatosues.blogspot.com/2011/06/bases-de-datos-distribuidas-con-mysql.html)

- MEREGHETTI Stefano. How-to – Install MySQL Cluster on Debian. {En línea}. {12 de Enero de 2015} Disponible en: (http://smereghetti.com/install-mysql-cluster-debian/)

- MySQL Cluster: Taller de bases de datos. {En línea}. {18 de Diciembre de 2014} Disponible en: (http://tavoberry.com/blog/mysql-cluster/)

- MySQL Cluster- Wikipedia, la enciclopedia libre. {En línea}. {15 de diciembre de 2014}. Disponible en: (http://es.wikipedia.org/wiki/MySQL_Cluster)

- MySQL: MySQL Cluster: Getting started. {En línea}. {15 de Diciembre de 2014} Disponible en: (http://www.mysql.com/products/cluster/start.html)

- MySQL: MySQL 5.1 Reference Manual. {En línea}. {15 de Marzo de 2015} Disponible en: (http://dev.mysql.com/doc/refman/5.1/en/mysql-cluster.html)

- MySQL:: MySQL 5.1 Reference Manual. Data Type Storage Requirements. {En línea} {25 de Febrero de 2015} (https://dev.mysql.com/doc/refman/5.1/en/storage-requirements.html)

- MySQL:: MySQL 5.1 Reference Manual. Limits and Differences of MySQL Cluster from Standard MySQL Limits. {En línea} {25 de Febrero de 2015} Disponible en: (https://dev.mysql.com/doc/refman/5.0/en/mysql-cluster-limitations-limits.html)

- NAVATHE Shamkant B., KARLAPALEM Kamalakar, RA Minyoung, “A Mixed Fragmentation Methodology For Initial Distributed Database Design”, Georgia Institute of Technology, Estados Unidos de América, Atlanta, Octubre de 2011.

- PÉREZ VALDÉZ Damián. ¿Qué son las bases de datos? {En línea} {20 de Diciembre de 2014}. Disponible en: (http://www.maestrosdelweb.com/editorial/%C2%BFque-son-las-bases-de-datos/)

- PETRINI Miercea, "Distribuited databases management using replication method", Revista "Economics", University of Petroşani, Rumania, Cluj-Napoca, Septiembre de 2009.

- Proxy- Wikipedia, la enciclopedia libre. {En línea}. {15 de Febrero de 2015} Disponible en: (http://es.wikipedia.org/wiki/Proxy)

- Reverse proxy- Wikipedia, the free encyclopedia. {En línea}. {15 de Febrero de 2015} Disponible en: (http://en.wikipedia.org/wiki/Reverse_proxy)

- SEGURA SANTIAGO Enrique de Jesús, Cuadro Comparativo entre Bases de Datos Centralizadas y Distribuidas. {En línea} {25 de Noviembre de 2014} Disponible en: (https://enriquesegsan.wordpress.com/2012/08/20/fundamentos-de-bases-de-datos/)

- SILBERSCHATZ Abraham, KORTH Henry F., SUDARSHAN S., "Fundamentos de bases de datos", cuarta edición, Madrid. McGRAW-HILL/INTERAMERICANA DE ESPAÑA, S. A. U., 2002.

- TAMER OZSU M., VALDURIEZ Patrick, Distributed Database Systems: Where Are We Now?

- TOTELA Eniko Elisabeta, COSTIN Aurelian Reazvan, "The desing of a distribuited database for doctoral studies management", Revista "Informatica Economică", Babes Bolyai University, Rumania, Cluj-Napoca, Mayo de 2010.

ANEXOS

Anexo A. Definiendo los nodos de MySQL Cluster.

https://dev.mysql.com/doc/refman/5.1/en/mysql-cluster-ndbd-definition.html#ndbparam-ndbd-datamemory

Anexo B. Limitaciones de MySQL Cluster.

https://dev.mysql.com/doc/refman/5.0/en/mysql-cluster-limitations-database-objects.html

Anexo C. Artículo científico.

PROTOTIPO DE UN CLÚSTER DE BASE DE DATOS HOMOGÉNEA DISTRIBUIDA EN ALTA DISPONIBILIDAD

Manuel Guillermo Flórez Becerra.
Head and Researcher of Conuss Group.
Professor at UIS University.
Bucaramanga, Colombia.
mgflorez@uis.edu.co

Daniel David De las Aguas Castellón
Researcher of Conuss Group.
Student at UIS University.
Bucaramanga, Colombia.
daniel.delasaguas@correo.uis.edu.co

Juan Carlos Acevedo Gutiérrez.
Researcher of Conuss Group.
Student at UIS University.
Bucaramanga, Colombia.
juan.acevedo3@correo.uis.edu.co

Abstract— MEIWEB is a virtual classroom that provides academic services to the Industrial University of Santander (UIS) using cloud technology, which involves handling a large amount of data stored in a centralized database; in order to optimize and increase the reliability and high availability of virtual classroom is necessary the existence of multiple databases synchronized in each of the virtual machines which reside MEIWEB.

To solve this problem it was decided to implement a homogeneous distributed databases system, which is a set of stored data synchronously with high availability. The difference of this system with the centralized system currently used is that these data are stored on different nodes in the case of the virtual classroom MEIWEB are virtual machines that make up a system and have connection with each other. Because the information is distributed nodes, query results can be obtained fast, flexible and reliable manner. Also, if any of the nodes fail, as all have local autonomy, the remaining active nodes continues to work without the system is turned off.

This research seeks to do a distributed database prototype (Cluster), which will undergo different tests to verify functionality and efficiency.

Keywords: Distributed databases, MySQL Cluster, replication.

I. INTRODUCCIÓN

Una base de datos distribuida (BDD) es una serie de múltiples bases de datos relacionadas entre sí, las cuales se encuentran distribuidas en diferentes espacios lógicos e intercomunicados mediante una red de comunicaciones, tienen total autonomía para realizar procesamientos locales y globales.

Entre las ventajas de utilizar bases de datos distribuidas encontramos una mayor tolerancia a fallos, lo que significa mayor confiabilidad, alta disponibilidad, mejor rendimiento, además de autonomía local. Sin embargo, la principal ventaja de este tipo de bases de datos es que simplifican las consultas a la base de datos.

Si bien el campo de los sistemas de bases de datos distribuidas es relativamente nuevo, algunas empresas de la industria automotriz, hotelera, financiera y de transporte aéreo, ya han empezado a migrar a éste sistema, debido a que es más compatible con su estructura o topología (Estas empresas suelen tener varias sucursales que comparten información). Además de que este sistema de bases de datos les supone una ventaja competitiva frente a los sistemas centralizados. Por otro lado, el sistema de base de datos que aún predomina es el sistema centralizado.

A pesar de su dominio en el mercado, las bases de datos centralizadas tienen ciertas debilidades que las hacen vulnerables a errores. Si el sistema de base de datos falla, se pierde la disponibilidad y procesamiento de la información que posee el sistema, y para su recuperación, dicho sistema no facilita la sincronización de todos los nodos, lo que permite irregularidades en los datos almacenados. Por otra

parte, no se puede hacer un balance en las cargas de trabajo entre las computadoras del sistema, lo que genera transacciones a baja velocidad y el desaprovechamiento de los recursos de las máquinas.

En el ámbito académico aún no se usan las bases de datos distribuidas, lo cual motiva la creación de un prototipo en el grupo de investigación GID-CONUSS de la escuela de Ingeniería de Sistemas e Informática de la UIS, en donde se implemente este sistema de bases de datos.

II. MARCO TEÓRICO

El concepto de una base de datos varía de acuerdo al contexto en que se utilice en el campo de la informática y especialmente en nuestro proyecto definiremos una base de datos (BD), como un conjunto de datos organizados y relacionados entre sí, los cuales son almacenados y utilizados por los sistemas de información de una empresa o entidad en particular.

Un sistema gestor de bases de datos (SGBD) consiste en un conjunto de datos interrelacionados y de programas para acceder a dichos datos. El principal objetivo de un SGBD es brindar mecanismos para almacenar y recuperar la información de una base de datos de manera tal que sea práctica y eficiente. [1]

Existen varios tipos de bases de datos; el presente proyecto se enfocará en las bases de datos distribuidas (BDD), definidas de la siguiente manera:

Existen algunas ventajas y desventajas de usar sistemas distribuidos de bases de datos tales como:

Tabla 1. Ventajas y desventajas de BDD.

Ventajas	Desventajas
Mayor rendimiento	Difícil control de concurrencias
Mayor confiabilidad	Mayor complejidad
Compartimiento de datos	Costos más altos
Alta disponibilidad de los datos Autonomía local	Aseguramiento de la integridad de los datos

El sistema de administración de base de datos distribuida (DDBMS) está formado por las transacciones y los administradores de base de datos distribuidos de todas las computadoras.

Un administrador de transacciones distribuidas (DTM) es un programa que se encarga de administrar las solicitudes de procesamiento de los programas de consulta o de transacciones y también las ejecuta para los DDBM.

El administrador de la base de datos (DBM) es un software que se encarga de procesar la base de datos distribuida (recuperar y actualizar datos del usuario) basándose en comandos de ejecución recibidos de los administradores de bases de datos.

Un nodo o sitio es una computadora con capacidad de ejecutar un DTM, un DBM, o ambos dependiendo del caso. Los DTM son procesados por nodos o sitios de transacción y a su vez, la base de datos y el DMB son procesados por el nodo de la base de datos. [3]

Tipos de bases de datos distribuidas

Dentro de las bases de datos distribuidas encontramos dos tipos: Las homogéneas y las heterogéneas. En el caso de estudio se utilizarán bases de datos distribuidas homogéneas ya que la base de datos de la plataforma MEIWEB es homogénea.

En las bases de datos distribuidas homogéneas los sitios tienen un software de sistemas gestores de bases de datos de características similares saben de la existencia de los demás sitios y están programados para responder entre todos las solicitudes de los usuarios. En este tipo sistemas los sitios locales dejan a un lado una parte de su autonomía, es decir, no pueden modificar los esquemas o el SGBD.

Hoy en día existen distintos productos comerciales con bases de datos distribuidas de varias empresas reconocidas, tales como: Relational Technology Inc, Oracle Corp e IBM. [2]

Por el lado de la academia existe software, entre ellos: Microsoft Access de Microsoft Corp, Oracle9i y MySQL. Este último recientemente lanzó su edición para BDD.

Cada uno de los anteriores tiene sus ventajas y sus desventajas.

En el caso de Microsoft Access, esta cuenta con un sistema gesto de bases de datos (SGBD) demasiado básico ya que está orientado a bases de datos muy poco sofisticadas, tampoco tiene en funcionamiento un sistema gestor de bases de datos distribuidas (SGBDD), pero tiene la ventaja de que se pueden implementar módulos que simulen la fragmentación, replicación y consultas distribuidas pero sin garantizar la fiabilidad que daría un SGBDD.

Oracle9i tiene un SGBD más escalable que Microsoft Access y también permite crear módulos que simulen fragmentación, replicación y consultas distribuidas.

MySQL ya cuenta con un SGBDD.

En bases de datos tradicionales, la redundancia se fue reduciendo por dos razones; la primera fue que la inconsistencia entre las diferentes copias de los mismos datos lógicos son automáticamente anuladas por tener sólo una copia, también que el espacio es salvado por la eliminación de la redundancia.

El grupo de investigación GID-CONUSS es el encargado del mantenimiento y optimización de la plataforma cloud-EISI la cual maneja un sistema de bases de datos centralizada. MEIWEB funciona en la nube, basándose en sistema de máquinas virtuales, y en una de ellas se encuentra

almacenada la base de datos, que a la vez, se encuentra conectada por medio de nodos a las otras máquinas virtuales, conformando un sistema de bases de datos centralizado.

Clúster de bases de datos

Un clúster es un conjunto de máquinas funcionando como unidad y trabajando juntas para tratar una única tarea, ahora bien, un MySQL Cluster se refiere a un grupo de máquinas trabajando juntas usando el motor de almacenamiento NDB ("Network Database") de MySQL para implementar almacenamiento de datos, recuperación y administración distribuida entre varias máquinas, lo que nos permite soportar una base de datos MySQL distribuida en una arquitectura de compartición nula usando almacenamiento en memoria.

Como mínimo se necesitan tres máquinas para la ejecución de un clúster, sin embargo, el número mínimo recomendado en MySQL Cluster es cuatro: una para el nodo de administración, otra para el de SQL, y dos para servir como nodos de almacenamiento. El propósito de los dos nodos de datos es proporcionar redundancia; el nodo de administración debe ejecutarse en una máquina separada para garantizar servicio de arbitraje continuo en caso que un nodo de datos falle.

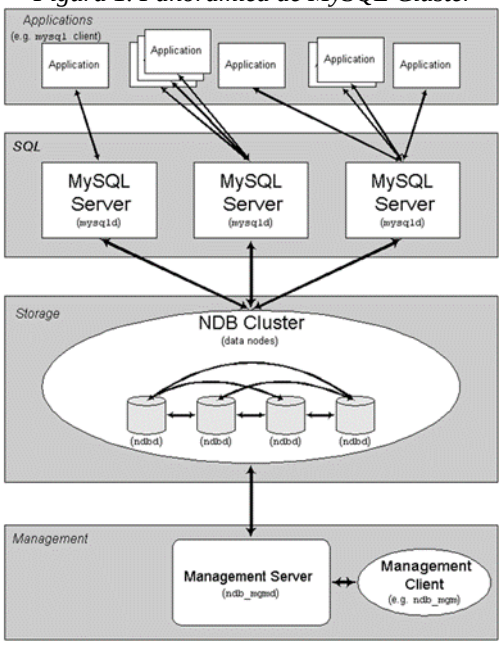
MySQL Cluster es una versión de alta disponibilidad, alta redundancia de MySQL ajustada para la computación distribuida. Usa el motor de almacenamiento ndbcluster para permitir la ejecución de varios servidores MySQL en un clúster. Este motor de almacenamiento está disponible en las distribuciones binarias de MySQL 5.0 y en los RPMs compatibles con las distribuciones Linux más modernas.

Lo sistemas operativos en que MySQL Cluster está disponible son Linux, Mac OS X, Solaris y Windows. [7]

Para esta investigación se utilizó en cada una de las máquinas virtuales, el sistema operativo Linux con distro debian Wheezy 7.7.0 con versión de kernel 3.2.0-4-486.

Un MySQL Cluster consiste en un conjunto de máquinas, cada una ejecutando un número de procesos incluyendo servidores MySQL, nodos de datos para ndbcluster, servidores de administración, y (posiblemente) programas especializados de acceso a datos. La relación de estos componentes en un clúster se muestra aquí:

Figura 1. Panorámica de MySQL Cluster



Fuente: Guebs [En línea] Disponible en: http://manuales.guebs.com/mysql-5.0/ndbcluster.html#mysql-cluster-overview

Todos estos programas funcionan juntos para formar un MySQL Cluster. Cuando se guardan los datos en el motor ndbcluster, las tablas se guardan en los nodos de datos. Tales tablas son automáticamente accesibles desde todos los otros servidores MySQL en el clúster.

CONCEPTOS BÁSICOS DE MYSQL CLUSTER

NDB: Motor de almacenamiento en memoria que ofrece alta disponibilidad y características de persistencia de datos.

Nodo: Es considerado como nodo, cada una de las partes de un MySQL Cluster.

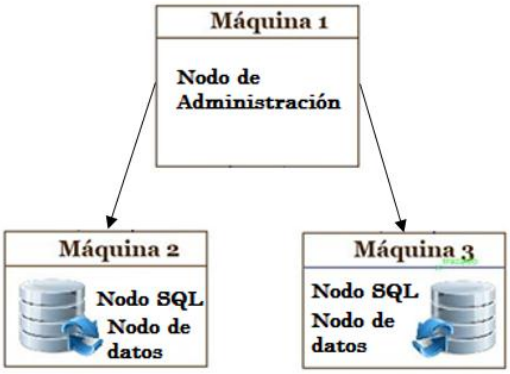
Existen tres tipos de nodos en un clúster, y en una configuración MySQL Cluster básica, mínimo habrá tres nodos, uno de cada tipo:

El nodo de administración (MGM): Este nodo es el encargado de administrar los otros nodos dentro del clúster, proporcionar datos de configuración, hacer copias de seguridad, arrancar y detener nodos. El nodo administrador debe ser el primero en ser iniciado. Un nodo MGM se arranca con el comando ndb_mgmd.

El nodo de datos: Es el encargado de almacenar los datos del clúster. En este proyecto se utilizaron dos nodos de este tipo. Un nodo de datos se arranca con el comando ndbd.

El nodo SQL: Este es el nodo que accede a los datos del clúster. En el caso de MySQL Cluster, un nodo cliente es un servidor MySQL tradicional que usa el motor ndbcluster. Un nodo SQL normalmente se arranca con el comando mysqld --ndbcluster o simplemente usando mysqld con ndbcluster añadido a my.cnf.

Figura 2. Nodos de un Clúster.



MySQLD: El proceso de servidor de bases de datos tradicional que puede ser utilizado en ambientes de clúster o aislado. Para que este proceso sea utilizado dentro de un clúster MySQL, necesita ser construido especialmente para soportar el motor de almacenamiento NDB, los binarios compilados disponibles en el sitio de MySQL ya integran esta funcionalidad al proceso.

NDBD: El proceso ndbd es el encargado de manejar todos los datos de las tablas utilizando el motor ndbd cluster. Este proceso soporta la funcionalidad de manejo de transacciones distribuidas entre los nodos, recuperación de nodos defectuosos o fuera de línea, checkpoint to disk (el momento en el que los datos son escritos efectivamente a disco), respaldo en línea, y otras tareas relativas a la distribución del clúster.

NDB_MGMD: Es el proceso que controla el servidor de administración, siendo responsable de conocer y mantener la configuración del clúster y distribuir dicha información a todos los nodos que la soliciten al unirse al clúster. Mantiene también el log de las actividades del clúster y reporta su estado a los clientes que se conectan a él.

Junto con el proceso NDB_MGMD se encuentra ndb_mgm, que es responsable de manejar el cliente de administración que interactúa con el nodo de administración. El cliente de administración se comunica con el nodo de administración utilizando una API en C. Esta API se puede utilizar para desarrollar aplicaciones dedicadas a controlar y administrar el clúster de una manera personalizada. [21]

BALANCEDOR DE CARGAS: Es un dispositivo de hardware o software que se usa para que un grupo de servidores atiendan una aplicación asignándoles o balanceando las solicitudes que llegan de los clientes a dichos servidores usando algoritmos, como por ejemplo, un simple Round Robin hasta otros de mayor sofisticación.

HAProxy: Es una solución de alta disponibilidad de tipo "open source" que proporciona equilibrio de carga y funciones proxy para aplicaciones basadas en TCP y HTTP mediante la difusión de las solicitudes a través de múltiples servidores.

Este es adecuado para sitios web de alto tráfico. Con el paso de los años HAProxy se ha convertido en el balanceador de carga de código abierto estándar, ahora trabaja con las principales distribuciones de Linux, y con frecuencia se despliega por defecto en plataforma en la nube.[8]

III. ESTADO DEL ARTE

GESTIÓN DE BASES DE DATOS DISTRIBUIDA USANDO MÉTODO DE REPLICACIÓN

DATOS DISTRIBUIDOS

Durante la década de los 70's la mayor parte de procesamiento de datos comerciales se produjo en sistemas informáticos grandes y centralizados.

REPLICACIÓN DE DATOS

Varios proveedores de DBMS han ido más allá de sus programas de extracción y de la utilidad de carga para ofrecer apoyo para la extracción de la tabla dentro del propio DBMS. Oracle, por ejemplo, dispone de vista materializada, una vista que almacena realmente las filas definidas por la consulta incluida en la definición de vista, para crear automáticamente una copia local de una tabla remota.

RÉPLICAS ACTUALIZABLES

En las implementaciones más simples, una mesa y sus réplicas tienen una estricta relación maestro / esclavo. La copia central / maestro contiene los datos reales. Es siempre al día, y todos los cambios a la tabla deben ocurrir en esta copia de la tabla. Las otras copias de esclavos están pobladas de actualizaciones periódicas, gestionados por el DBMS. Las aplicaciones que exigen el uso de alta disponibilidad replican tablas para mantener copias idénticas de los datos en dos sistemas informáticos diferentes. Si un sistema falla, el otro contiene los datos actuales y puede continuar su procesamiento. Una aplicación de Internet puede exigir tasas muy altas de acceso de base de datos. [19]

ARQUITECTURA DE REFERENCIA PARA LAS BASES DE DATOS DISTRIBUIDAS

Los elementos que forman la arquitectura son los siguientes:

Esquema global: Define todos los datos que están contenidos en la BDD como si la BD no fuese distribuida.

Fragmentos: Cada relación global puede ser dividida en porciones que no se solapen, llamados fragmentos. El mapa resultante se denomina esquema de fragmentación.

FRAGMENTACIÓN

- **Fragmentación horizontal**

La fragmentación horizontal se realiza sobre las tuplas de la relación. Cada fragmento será un subconjunto de las tuplas de la relación. Existen dos variantes de la fragmentación

horizontal: La fragmentación horizontal primaria de una relación se desarrolla empleando los predicados definidos en esa relación. Por el contrario, la fragmentación horizontal derivada consiste en dividir una relación partiendo de los predicados definidos sobre alguna otra.

- **Fragmentación vertical**

El objetivo de la fragmentación vertical es particionar una relación en un conjunto de relaciones más pequeñas de manera que varias de las aplicaciones de usuario se ejecutarán sobre un fragmento.

- **Fragmentación mixta o híbrida**

La forma más simple de construir fragmentación mixta consiste en aplicar fragmentación horizontal a fragmentos verticales, o aplicar fragmentación vertical a fragmentos horizontales.

PROBLEMAS DE LA FRAGMENTACIÓN

Encontrar la unidad apropiada de distribución, es decir, definir qué contiene un fragmento.

El rendimiento se afecta cuando existen aplicaciones que necesitan tener una vista completa de un objeto o entidad (relación, en el modelo relacional) y está descompuesta en fragmentos ubicados físicamente en sitios distintos.

Se pierde el significado semántico del objeto o entidad al tenerse el concepto subdivido en fragmentos ubicados en diferentes sitios.

A pesar de estos inconvenientes, la fragmentación facilita el proceso concurrente de las transacciones y, por lo tanto, la recuperación de información.

ALTERNATIVAS SOBRE REPLICACIÓN PARA LA ASIGNACIÓN DE FRAGMENTOS

Si la fragmentación es correcta se decide cómo asignar tales fragmentos a los diferentes sitios de la red. En el caso en que una serie o grupo de datos sean asignados para la fragmentación, pueden ser replicados para mantener una copia. Una base de datos fragmentada es aquella donde no existe réplica alguna. Los fragmentos de la base de datos son almacenados en sitios en los cuales existe solamente una copia de cada uno de ellos en toda la red. En caso de réplica, se puede considerar una base de datos totalmente replicada, donde existe una copia de todo el banco de datos en cada sitio, o considerar una base de datos parcialmente replicada donde existan copias de los fragmentos ubicados en diferentes sitios. [20]

IV. IMPLEMENTACIÓN

Configuración del clúster de MySQL

Para iniciar la configuración del clúster se debe instalar linux en máquinas virtuales, las cuales serán los nodos que conforman el clúster. Para este caso específico se utilizaron 4 máquinas virtuales con un distro debian Wheezy 7.7.0.

Como primera medida se debe crear una red interna en donde se asignan direcciones IP a cada una de las máquinas para establecer comunicación entre los nodos del clúster.

Para empezar con la configuración del clúster de MySQL para 4 nodos: Dos nodos administradores (“maquina1” y “maquina4”) y dos nodos de datos (“maquina2” y “maquina3”). Lo primero que se debe hacer es descargar el MySQL server en cada uno de los nodos. MySQL Server se compone de tres paquetes: *mysql-common*, *mysql-client* y *mysql-server*.

Después de hacer la instalación de *mysql server* se procede a configurar MySQL Cluster en cada uno de los nodos, empezando por los nodos administradores, para lo cual debemos descargar el paquete de instalación de MySQL desde su página oficial dependiendo de la arquitectura del pc.

Configuración de los nodos de administración

Para la iniciación de los servicios de MySQL Cluster se deben crear archivos de configuración para indicarle al programa cuáles son los nodos y qué función cumplen además de indicar en dónde se guardarán los datos. El primer archivo de configuración se llamará config.ini, éste estará en *“/var/lib/mysql-cluster”* directorio que debe ser creado.

Configuración de los nodos de almacenamiento

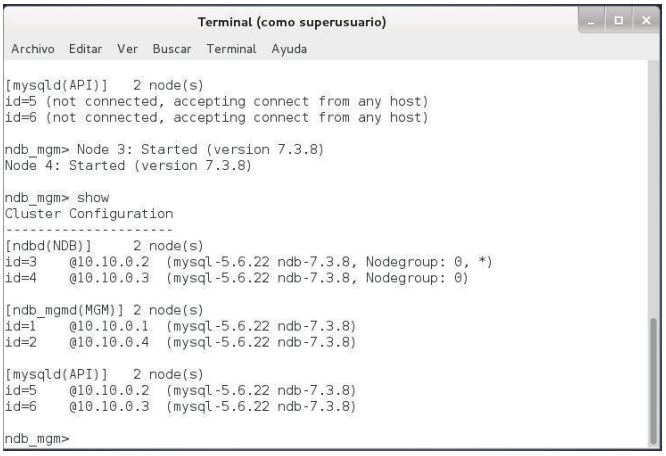
1. Se debe crear un grupo MySQL y crear un usuario, esto con los siguientes comandos *“groupadd mysql”* y *“useradd -g mysql mysql”.*
2. En los nodos de almacenamiento, en el directorio /etc/ creamos en cada nodo un archivo llamado “my.cnf”, donde irán los datos de conexión de cada una de dichas máquinas.
3. Se debe hacer lo anterior en ambos nodos. Se crea un enlace simbólico llamado “*mysql*” desde el archivo “*mysql-cluster*” que se descargó anteriormente en cada una de las máquinas.
4. Se ubica el directorio con el enlace simbólico y se instalan actualizaciones del kernel del sistema operativo y además se instalan las bases de datos por defecto que tiene MySQL bajo el usuario “mysql” creado anteriormente y se le asigna un directorio para el almacenamiento de los datos.

Inicio de los servicios de MySQL Cluster

Para iniciar la operación del clúster es necesario iniciar los servicios de MySQL Cluster respectivos en cada uno de los nodos estableciendo un orden para evitar la generación de errores.

1. Se debe iniciar el servicio de los nodos de administración.
2. Se inicia el servicio del demonio ndbd en cada una de las máquinas.
3. Iniciar el servicio de mysql server. Este paso se debe hacer necesariamente en este orden.

El demonio del nodo SQL es el que se encargará de hacer la replicación de los datos ingresados a la base de datos.
Desde el nodo administrador se puede ver que todos los nodos están conectados de manera correcta.
Imagen 1. Verificación de inicio de los servicios.



Instalación y configuración del HAProxy

En esta investigación se instaló HAProxy en los nodos administradores, para que balanceen la carga de las bases de datos.

Primero se deben preparar los servidores MySQL mediante la creación de dos usuarios adicionales para HAProxy (haproxy_check y haproxy_root). El primer usuario será utilizado por HAProxy para comprobar el estado de un servidor y el otro para direccionar las peticiones.
MySQL-Client tiene que estar instalado en el nodo de HAProxy para probar la conectividad.
Para instalar HAProxy se necesita habilitar el repositorio backports.

Configuración del HAProxy

Se debe editar el archivo de configuración de HAProxy (haproxy.cfg).

Pruebas y verificación:

Para comenzar se debe ingresar a la api de *mysql* desde la cual se realizarán todas las operaciones pertinentes de la base de datos.

MySQL por defecto tiene varias bases de datos creadas. Para la prueba fue necesario crear una base de datos con el nombre de ‘’agenda’’.
Nota: Es de suma importancia que el “ENGINE” sea *ndbcluster*, debido a que si utiliza otro valor como por ejemplo INNODB, no habrá replicación de los datos entre los nodos del clúster.

PRUEBA LOCAL

Se agregaron, eliminaron y modificaron registros en la base de datos desde uno de los nodos SQL y se comprobó que estos registros se replicaban en el otro nodo SQL.
Simulando un gestor de archivos
Además de la prueba anterior fue creada una tabla desde la cual se hacen inserciones de archivos por medio de una aplicación web. Esta tabla guarda los archivos en una carpeta de la misma máquina en donde se aloja la aplicación, ahí guarda la ruta de dicho directorio. Cuando se hizo la inserción de los archivos, el clúster fue sometido a todas las operaciones de la prueba anterior para comprobar la alta disponibilidad de los datos y el correcto funcionamiento de la herramienta.

ELIMINACIÓN E INSERCIÓN DE DATOS CON MÁQUINAS INACTIVAS

Simulando la caída de uno de los nodos SQL del clúster
Se agregaron y eliminaron registros desde el nodo SQL activo. Se observó que el clúster sigue funcionando y además que al reconectar el nodo inactivo, se replican todas las transacciones hechas desde el otro nodo.
Simulando la caída de uno de los nodos de administración La prueba consistió en desconectar uno de los nodos administradores y realizar las operaciones de la prueba anterior. Se observó que el clúster sigue funcionando.

V. CONCLUSIONES

Se realizó un prototipo de bases de datos distribuido con MySQL Cluster, que se implementó en la nube “cloud-EISI” del grupo de investigación GID-CONUSS, dicho prototipo fue sometido a distintas pruebas, entre esas se encuentran las de funcionalidad, integridad de datos, mantenibilidad de transacciones y tolerancia a fallos, lo que genera una mayor confiabilidad y alta disponibilidad de la base de datos.

La implementación de este nuevo sistema de bases de datos supone un nuevo desafío para el grupo de investigación GID-CONUSS ya que el sistema ya mencionado requiere personal con mejores capacidades para el mantenimiento y la administración del sitio, además de tener visión de futuras mejoras.

VI. RECOMENDACIONES

Asignar la mayor cantidad de espacio en memoria (RAM) posible a las máquinas virtuales para evitar problemas al cargar datos en el clúster.

Utilizar imagen del sistema operativo usado (en este caso Debian) sin interfaz gráfica, para optimizar la RAM de las máquinas virtuales.

Basados en los resultados obtenidos en este proyecto, no es recomendable alojar bases de datos relativamente grandes en el clúster, debido a las limitaciones que presenta MySQL Cluster.

En el momento de crear los usuarios de mysql que accederán a las bases de datos requeridas, es de suma importancia otorgarles todos los permisos y privilegios posibles dichos usuarios sin que se afecte el sistema, con el fin de que no se presenten inconvenientes en el manejo de las bases de datos.

Una de las funcionalidades más importantes de un sistema de bases de datos distribuidas, es que debe funcionar independientemente de la red a la cuál esté conectado, por eso es indispensable configurar una red interna para que se comuniquen entre sí los nodos que hacen parte del clúster.

En caso de necesitar otro nodo en el clúster, se recomienda hacer una copia de las bases de datos que haya, porque es necesario reiniciar completamente el servicio y cambiar la configuración para agregar el nuevo nodo, de esta manera se evita que haya pérdida de información.

Se recomienda para futuras investigaciones en este tema, consultar otras herramientas utilizadas para la configuración y administración de bases de datos distribuidas. Actualmente además de la herramienta utilizada en este proyecto se encuentran otras libres, tales como: Galera Cluster, Percona XtraDB Cluster, MariaDB Galera Cluster.

REFERENCIAS

[1] Abraham Silberschatz, Henry F. Korth, S. Sudarshan, “Fundamentos de bases de datos”, cuarta edición, McGRAW-HILL/INTERAMERICANA DE ESPAÑA, S. A. U., 2002.

[2] Carmen Costilla, “Arquitecturas de los SGBD distribuidos”, Departamento de Ingeniería de Sistemas Telemáticos, Universidad Politécnica de Madrid, Madrid, Diciembre de 2009.

[3] Departamento de Ingeniería de Sistemas, maestría en ciencias de la computación, Universidad de Colima, “Sistemas de bases de datos distribuidas”.

[4] Distributed Database Systems: Where Are We Now? M. Tamer Ozsu, GTE Laboratories* Patrick Valduriez, INRIA

[5] DISEÑO DE UN SISTEMA GESTOR DE BASE DE DATOS DISTRIBUIDA BASADO EN ORACLE9i, Celia Gutiérrez Cosío, Revista “Tecnología y desarrollo”, Universidad Alfonso X El Sabio, Madrid 2008.

[6] Bases de datos de la Universidad de El Salvador: Bases de datos distribuidas con MySQL. [En línea] Disponible en: http://basesdedatosues.blogspot.com/2011/06/bases-de-datos-distribuidas-con-mysql.html

[7] Capítulo 16. MySQL Cluster [En línea] Disponible en: http://manuales.guebs.com/mysql-5.0/ndbcluster.html#multi-config

[8] Reverse proxy- Wikipedia, the free encyclopedia. [En línea] Disponible en: http://en.wikipedia.org/wiki/Reverse_proxy

[9] MySQL:: MySQL 5.1 Reference Manual. [En línea] Disponible en: http://dev.mysql.com/doc/refman/5.1/en/mysql-cluster.html

[10] How-to – Install MySQL Cluster on Debian. [En línea] Disponible en: http://smereghetti.com/install-mysql-cluster-debian/

[11] How To Use HAProxy to Set Up MySQL Load Balancing. [En línea] Disponible en: https://www.digitalocean.com/community/tutorials/how-to-use-haproxy-to-set-up-mysql-load-balancing--3

[12] Instalación, Configuración y demostración de un MySQL Cluster. [En línea] Disponible en: http://informacion-activa.blogspot.com/2012/04/instalacion-configuracion-y.html

[13] Instalación de MySQL Cluster (ndb). [En línea] Disponible en: http://systemadmin.es/2012/01/instalacion-de-mysql-cluster-ndb

[14] Montar un cluster de MySQL. [En línea] Disponible en: http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=MySQLCluster

[15] How to setup mysql-cluster (beginners tutorial). [En línea] Disponible en: https://cyrenity.wordpress.com/2010/08/12/howto-setup-mysql-cluster-beginners-tutorial/

[16] MySQL with HAProxy for Failover and Load Balancing. [En línea] Disponible en: http://www.jamesattard.com/2014/09/mysql-with-haproxy-for-failover-and.html

[17] MySQL:: MySQL 5.1 Reference Manual. Limits and Differences of MySQL Cluster from Standard MySQL Limits. [En línea] Disponible en:

https://dev.mysql.com/doc/refman/5.0/en/mysql-cluster-limitations-limits.html

[18] MySQL:: MySQL 5.1 Reference Manual. Data Type Storage Requirements. https://dev.mysql.com/doc/refman/5.1/en/storage-requirements.html

[19] Miercea Petrini, "Distribuited databases management using replication method", University of Petroşani, Rumania, Economics, vol. 9, no. 4. 2009.

[20] Eniko Elisabeta TOTELA, Aurelian Reazvan COSTIN, "The desing of a distribuited database for doctoral studies management", Babes Bolyai University, Cluj-Napoca, Rumania, Informatica Economică vol. 14, no. 4/2010.

MySQL Cluster- Wikipedia, la encyclopedia libre [En línea] Disponible en: http://es.wikipedia.org/wiki/MySQL_Cluster