UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE ESTUDIOS SUPERIORES CUAUTITLÁN

IMPLEMENTACIÓN DE UN CLUSTER DE ALTA DISPONIBILIDAD Y BALANCEO DE CARGAS PARA LA PROTECCIÓN DE LA INFORMACIÓN EN LA FES CUAUTITLÁN.

T E S I S

QUE PARA OBTENER EL TÍTULO DE:

LICENCIADO EN INFORMÁTICA

PRESENTAN:
JOSÉ DANIEL SALGADO RODRÍGUEZ
URIEL EDMUNDO TOVAR COLÍN

ASESOR: LEONEL GUALBERTO LÓPEZ SALAZAR

CUAUTITLÁN IZCALLI, ESTADO DE MÉXICO 2015

VNIVERSIDAD NACIONAL AVPNOMA DE MEXICO

FACULTAD DE ESTUDIOS SUPERIORES CUAUTITLÁN
UNIDAD DE ADMINISTRACIÓN ESCOLAR
DEPARTAMENTO DE EXÁMENES PROFESIONALES

U. N. A. M.
FACULTAD DE ESTUDIOS SUPERIORES CUAUTITLAN

ASUNTO: VOTO APROBATORIO

M. en C. JORGE ALFREDO CUÉLLAR ORDAZ
DIRECTOR DE LA FES CUAUTITLAN
PRESENTE

ATN: M. en A. ISMAEL HERNANDEZ MAURICIO
Jefe del Departamento de Exámenes
Profesionales de la FES Cuautitlán.

DEPARTAMENTO DE EXAMENES PROFES.

Con base en el Reglamento General de Exámenes, y la Dirección de la Facultad, nos permitimos comunicar a usted que revisamos **La Tesis:**

Implementación de un cluster de alta disponibilidad y balanceo de cargas para la protección de la información en la FES Cuautitlán

Que presenta el pasante: **JOSÉ DANIEL SALGADO RODRÍGUEZ**
Con número de cuenta: **30630244-2** para obtener el Título de: **Licenciado en Informática**

Considerando que dicho trabajo reúne los requisitos necesarios para ser discutido en el **EXAMEN PROFESIONAL** correspondiente, otorgamos nuestro **VOTO APROBATORIO**

ATENTAMENTE
"POR MIRAZA HABLARÁ EL ESPÍRITU"
Cuautitlán Izcalli, Méx. a 11 de junio del 2015.

PROFESORES QUE INTEGRAN EL JURADO

	NOMBRE	FIRMA
PRESIDENTE	M.I. Juan Carlos Axotla García	
VOCAL	L.R.I. Verónica Cuadros Jiménez	
SECRETARIO	M.G.T.I. Leonel Gualberto López Salazar	
1er SUPLENTE	L.I. Oscar Vilchis Guerra	
2do SUPLENTE	L.I. Rosalba Nancy Rosas Fonseca	

VNIVERSIDAD NACIONAL AVPNOMA DE MEXICO

FACULTAD DE ESTUDIOS SUPERIORES CUAUTITLÁN
UNIDAD DE ADMINISTRACIÓN ESCOLAR
DEPARTAMENTO DE EXÁMENES PROFESIONALES

U. N. A. M.
FACULTAD DE ESTUDIOS SUPERIORES CUAUTITLAN

ASUNTO: VOTO APROBATORIO

M. en C. JORGE ALFREDO CUÉLLAR ORDAZ
DIRECTOR DE LA FES CUAUTITLAN
PRESENTE

ATN: M. en A. ISMAEL HERNANDEZ MAURICIO
Jefe del Departamento de Exámenes
Profesionales de la FES Cuautitlán.

DEPARTAMENTO DE EXAMENES PROFES. FES

Con base en el Reglamento General de Exámenes, y la Dirección de la Facultad, nos permitimos comunicar a usted que revisamos **La Tesis:**

Implementación de un cluster de alta disponibilidad y balanceo de cargas para la protección de la información en la FES Cuautitlán

Que presenta el pasante: **URIEL EDMUNDO TOVAR COLÍN**
Con número de cuenta: **40909435-8** para obtener el Título de: **Licenciado en Informática**

Considerando que dicho trabajo reúne los requisitos necesarios para ser discutido en el **EXAMEN PROFESIONAL** correspondiente, otorgamos nuestro **VOTO APROBATORIO**

ATENTAMENTE
"POR MIRAZA HABLARÁ EL ESPÍRITU"
Cuautitlán Izcalli, Méx. a 11 de junio del 2015.

PROFESORES QUE INTEGRAN EL JURADO

	NOMBRE	FIRMA
PRESIDENTE	M.I. Juan Carlos Axotla García	
VOCAL	L.R.I. Verónica Cuadros Jiménez	
SECRETARIO	M.G.T.I. Leonel Gualberto López Salazar	
1er SUPLENTE	L.I. Oscar Vilchis Guerra	
2do SUPLENTE	L.I. Rosalba Nancy Rosas Fonseca	

Dedicatoria José Daniel

A mis Padres y familia, por su apoyo incondicional, amor y ayuda durante toda mi formación académica.

A la UNAM, por darme la oportunidad de desarrollarme en todos los aspectos de mi vida, por acercarme al Software libre e iniciar mí gusto por Linux.

Al Prof. José Juan Rico, por todos sus consejos sobre LINUX, los interesantes proyectos del Servicio Social que sirvieron como ideas para esta tesis.

Al Prof. Leonel López por ser una excelente persona, brindarnos su ayuda incondicional, confianza y apoyo en la elaboración y revisión de esta tesis.

A mi compañero Uriel, por su gusto a la Tecnología, ayuda y compromiso durante la realización de esta tesis y sobre todo por ser un excelente amigo.

A Liz, por ser el amor de mi vida, darme los mejores consejos, ayuda y apoyo incondicional durante ya más de 8 años.

Dedicatoria Uriel Edmundo

A mis Padres y familia, por su apoyo, amor, trabajo y sacrificio durante toda mi formación académica.

A la UNAM, por darme la oportunidad de desarrollarme personal y profesionalmente.

Al Profesor José Juan Rico, por sus enseñanzas y consejos sobre Linux.

Al Profesor Leonel López, por su ayuda incondicional y guiarnos de principio a fin en el elaboración de ésta tesis.

A mi compañero Daniel, por su pasión por el software libre, por compartir día a día algo nuevo sobre Linux y sobre todo por brindarme su amistad.

ÍNDICE

Introducción

Con la proliferación de Internet, las aplicaciones de misión crítica que procesan y almacenan datos sensibles son muy comunes dentro de las organizaciones, para las cuales es inaceptable que se presenten fallas o un rendimiento ineficiente de dichas aplicaciones que puede repercutir en la pérdida de información. El balanceo de carga y la alta disponibilidad son una solución muy aceptable para enfrentar el reto de proporcionar servicios de red continuos y eficientes para los usuarios.

Actualmente las soluciones existentes de Balanceo de Carga y Alta Disponibilidad en organizaciones donde el flujo de datos es constante y demandante representa un costo muy elevado, el cual no todas las empresas pueden pagarlo y por lo tanto sus sistemas comienzan a presentar cargas de trabajo en horas pico lo que lleva a la interrupción o disminución del servicio, y representa pérdidas para la organización.

Éste trabajo, propone contar con un diseño adecuado de los datos y garantizar la disponibilidad de los contenidos y servicios web de la FES-Cuautitlán, como una solución a los problemas que se presentan actualmente debido al aumento de la demanda de usuarios a los servidores así como minimizar afectación a los sistemas frente a posibles ataques de denegación de servicio (DoS).

Se desarrollará una arquitectura de Alta Disponibilidad usando Software Libre, en específico bajo la plataforma LINUX y equipos de cómputo ya disponible dentro de la Facultad, por lo que la elaboración del proyecto no conllevara costo alguno, así mismo la aplicación web a la que se le proporcionara ese servicio será al ambiente educativo Virtual Moodle ya que es ampliamente usada por los docentes y cuenta con las características estándar de las aplicaciones que se desarrollan o usan dentro de la facultad lo que provocara que se pueda replicar el proyecto a otras aplicaciones web que se usan en la Facultad.

Implementación de un cluster de alta disponibilidad y balanceo de cargas para la protección de la información en la FES Cuautitlán

Objetivos

General:

Diseñar e implementar una arquitectura basada en un cluster que permita proteger la información que se almacena en los servidores del centro de cómputo de la FES-Cuautitlán.

Específicos:

- Ofrecer un respaldo de la información en tiempo real en caso de que se presente un fallo físico en el servidor
- Implementar el servicio de balanceo de carga para protección de caídas por exceso tráfico.
- Diseñar scripts de configuración que simplifiquen el proceso de instalación y configuración de un cluster de alta disponibilidad y balanceo de carga para usos futuros**.**

Hipótesis:

Hipótesis Nula. Al diseñar e implementar un cluster de Alta disponibilidad y Balanceo de carga en el centro de cómputo de la FES-Cuautitlán, no se lograra una diferencia significativa en la eliminación de ataques, fallas y contingencias.

Hipótesis Alternativa. Al diseñar e implementar un cluster de Alta disponibilidad y Balanceo de carga en el centro de cómputo de la FES Cuautitlán, se lograra una diferencia significativa en la eliminación de ataques, fallas y contingencias.

ANTECEDENTES

Antes de que existiera la tecnología de cluster, la única forma de trabajar en la solución de problemas que requerían demasiados recursos computacionales, era invertir en equipo con mejores características y esperar que el tiempo de obsolescencia de estos no fuera muy corto. En los últimos años los cluster han tomado fuerza tanto en las universidades como en las organizaciones, debido a que es una forma económica de solucionar problemas de rendimiento y disponibilidad en aplicaciones críticas.

Dentro de la Facultad de Estudios Superiores Cuautitlán no se ha implementado algún sistema de Alta Disponibilidad, lo que ha representado perdida de información y un servicio lento en el momento en que los alumnos generan cargas excesivas en los sistemas. El respaldo básico de la información es el único medio de protección con el que se cuenta.

Justificación

El cómputo con clusters surge como resultado de varias tendencias actuales que incluyen la disponibilidad de microprocesadores económicos de alto rendimiento y redes de alta velocidad, el desarrollo de herramientas de software para cómputo distribuido de alto rendimiento, así como la creciente necesidad de recursos para aplicaciones que la requieran.

El uso del Software libre en la elaboración de soluciones dentro de una organización trae como consecuencia que sea posible adaptar el software al problema específico que se desea resolver y con esto facilitar las adaptación a cambios futuros de la infraestructura de computo.

CAPÍTULO I INFRAESTRUCTURA DE TI

1.1 Modelo OSI

El modelo de Interconexión de Sistemas Abiertos (OSI: Open System Interconection) fue diseñado por la Organización para la Estandarización Internacional (ISO: International Standard Organization), éste pretendía ser un modelo de referencia que permitiera que dos sistemas distintos se comuniquen sin importar su arquitectura.

Definición

El modelo básico de referencia OSI, o simplemente modelo OSI, afronta el problema de las telecomunicaciones de datos y las redes informáticas dividiéndolo en niveles. Cada participante de la comunicación incorpora como mínimo uno de los mismos, y los equipos terminales los incorporan todos[1].

El modelo en sí mismo no puede ser considerado una arquitectura, ya que no especifica el protocolo que debe ser usado en cada capa, sino que suele hablarse de modelo de referencia.

Este modelo está dividido en siete capas:

Capa 1.- Nivel físico

Se encarga de coordinar las funciones necesarias para transmitir el flujo de datos a través de un medio físico, trata con las especificaciones eléctricas y mecánicas de la interfaz y del medio de transmisión. El nivel físico se relaciona con lo siguiente:

- Características físicas de las interfaces y el medio.- Define el tipo de medio de transmisión, las características de la interfaz entre dispositivos y el medio de

[1] Estructura de redes de computadores, Jordi Íñigo Griera, José M. Barceló Ordinas

transmisión.

- Representación de los bits.- Los datos del nivel físico están compuestos por un flujo de bits sin ninguna interpretación, para que puedan ser transmitidos es necesario codificarlos en señales eléctricas u ópticas. El nivel físico define el tipo de codificación.
- Tasa de datos.- El nivel físico también define la tasa de transmisión: el número de bits enviados en cada segundo, es decir la duración del bit
- Sincronización de los bits.- El emisor y el receptor deben estar sincronizados a nivel de bit.
- Configuración de la línea.- El nivel físico está relacionado con la conexión del dispositivo al medio. En una configuración punto a punto se conectan dos dispositivos a través de un enlace dedicado, si la configuración es multipunto, el enlace es compartido.
- Topología física.- Define cómo están conectados todos los dispositivos para formar una red.
- Modo de transmisión.- El nivel físico también define el modo de transmisión entre dispositivos, simplex, half dúplex o full dúplex.

Capa 2.- Nivel de enlace de datos.

Transforma el nivel físico, el medio de transmisión, en un enlace fiable y es responsable de la entrega nodo a nodo, y hace que el nivel físico aparezca ante el nivel superior (nivel de red) como un medio libre de errores. Las responsabilidades específicas de este nivel son:

- Tramado. El nivel de enlace divide el flujo de bits recibidos del nivel de red en unidades de datos manejables denominados tramas.
- Direccionamiento físico. Si es necesario distribuir las tramas por distintos sistemas de la red, el nivel de enlace añade una cabecera para definir la dirección física del emisor (dirección fuente) y/o receptor (dirección destino) de la trama. Si hay que enviar la trama a un sistema fuera de la red del emisor, la dirección del receptor es la del dispositivo que conecta las dos redes.

- Control del flujo. Si la velocidad a la que el receptor recibe los datos es menor que la velocidad de transmisión del emisor, el nivel de enlace impone un mecanismo de control del flujo de datos para prevenir el desbordamiento del receptor.
- Control de errores. El nivel de enlace añade fiabilidad al nivel físico al incluir mecanismos para detectar y retransmitir las tramas defectuosas o perdidas. También usa un mecanismo para prevenir la duplicidad de tramas. Usualmente ese control se realiza a través de una cola que se añade al final de la trama.
- Control de acceso. Cuando se conectan dos o más dispositivos al mismo enlace, los protocolos de nivel de enlace deben determinar en todo momento qué dispositivo tiene el control del enlace.

Capa 3.- Nivel de red

El nivel de red es responsable de la entrega de un paquete desde el origen al destino y, posiblemente, a través de múltiples redes. El nivel anterior supervisa la entrega del paquete de datos, el nivel de red asegura que cada paquete va del origen al destino, sean estos cuales sean. Si dos sistemas están conectados al mismo enlace, habitualmente no hay necesidad de esta capa, sin embargo si los dos sistemas están conectados a redes distintas es necesario un nivel de red para llevar a cabo la entrega. Las responsabilidades de esta capa son:

- Direccionamiento lógico. El direccionamiento físico del nivel de enlace gestiona los problemas de direcciones locales (en la misma red). Si un paquete cruza la frontera de la red, es necesario tener otro tipo de direcciones para distinguir los sistemas origen de los de destino. El nivel de red añade una cabecera al paquete que viene del nivel superior que, entre otras cosas, incluye las direcciones lógicas del emisor y el receptor.
- Enrutamiento (routing). Cuando un conjunto de redes o enlaces independientes se conectan juntas para crear una red de redes o una red más grande, los dispositivos de conexión denominados encaminadores o pasarelas, encaminan los paquetes hasta su destino final.
- Reenvío (forwarding). Cuando un paquete llega por el enlace de entrada de un router,

este tiene que pasar el paquete por el enlace de salida apropiado, utilizando para ello dos interfaces una de entrada, una de salida y sus tablas de reenvío, router o equipo de interconexión puede llegar a tener más de dos interfaces ya se apara la entrada o para la salida ya las tablas de reenvío o encaminamiento que contienen la información indexada, asociada a la salida de cada interfaz.

Capa 4.- Nivel de transporte

El nivel de transporte es responsable de la entrega origen a destino (extremo a extremo) de todo el mensaje. Mientras que el nivel de red supervisa la entrega de paquetes individuales, no reconoce ninguna relación entre ellos, tratando cada uno independientemente de los demás. El nivel de transporte asegura que todo el mensaje llega intacto y en orden supervisando tanto el control de errores como el control de flujo a nivel origen a destino. Para una mayor seguridad en nivel de transporte puede crear una conexión entre dos puertos finales, una conexión es un único camino entre el origen y el destino asociados a todos los paquetes del mensaje. Algunas de las responsabilidades de este nivel son:

- Direccionamiento en punto de servicio. Dado que el sistema suele ejecutar varios programas al mismo tiempo, la entrega del origen al destino significa la entrega a un proceso específico del sistema destino. La cabecera del nivel de transporte debe además incluir un tipo de dirección denominado dirección de punto de servicio (o dirección del puerto).
- Segmentación y reensamblado. Un mensaje se divide en segmentos transmisibles, cada uno de los cuales contiene un cierto número de secuencias. Estos números permiten al nivel de transporte reensamblar el mensaje correctamente a su llegada a destino e identificar y reemplazar paquetes perdidos en la transmisión.
- Control de conexión. El nivel de transporte puede estar orientado a conexión o no. Si no está orientado a conexión, trata cada segmento como un paquete independiente y lo pasa al nivel de transporte del destino. Si está orientado a conexión establece una conexión con el nivel de red de destino antes de enviar ningún paquete, la conexión se corta después de que se han transferido todos los paquetes.

- Control de flujo. Al igual que el nivel de enlace, el nivel de transporte es responsable del control de flujo, que se lleva a cabo de extremo a extremo y no sólo en un único enlace.
- Control de errores. El nivel de transporte es responsable del control de errores de extremo a extremo y no sólo en un único enlace (envío de un paquete). El nivel de transporte del emisor asegura que todo el mensaje llega al nivel de transporte del receptor sin errores (daños, pérdidas o duplicaciones). Habitualmente los errores se corrigen mediante retransmisiones.

Capa 5.- Nivel de sesión

Los servicios provistos por los tres primeros niveles no son suficientes para algunos procesos, el nivel de sesión es el controlador de diálogo de la red, establece, mantiene y sincroniza la interacción entre sistemas de comunicación. Entre sus responsabilidades están:

- Control de diálogo. El nivel de sesión permite que dos sistemas establezcan un diálogo, permite que la comunicación de dos procesos tenga lugar en modo simplex, half dúplex o full duplex.
- Sincronización. El nivel de sesión permite que un proceso pueda añadir puntos de prueba en el flujo de datos, para asegurar una recepción parcial de los datos.

Capa 6.- Nivel de presentación

El nivel de presentación está relacionado con la sintaxis y la semántica de la información intercambiada entre dos sistemas. Las responsabilidades de este nivel son:

- Traducción. Los procesos en los sistemas intercambian la información en forma de tiras de datos (números, caracteres, etc.), es necesario traducir la información a flujos de bits antes de transmitirla. Debido a que cada proceso (programa) o cada sistema usa un sistema de codificación distinto, el nivel de presentación es responsable de la

interoperatividad entre los distintos sistemas de codificación, cambiando la información del formato dependiente del emisor a un formato común. El nivel de presentación del receptor cambia el formato común en formato específico del receptor.

- Cifrado. Para transportar información sensible, el sistema debe asegurar la privacidad. El cifrado implica que el emisor transforma la información original a otro formato, normalmente codificación de seguridad, y envía el mensaje resultante por la red. El nivel de presentación del receptor descifra el mensaje para convertirlo al formato original.
- Compresión. La compresión reduce el número de bits a transmitir, lo cual es particularmente importante en la transmisión de datos multimedia.

Capa 7.- Nivel de aplicación

El nivel de aplicación permite al usuario, tanto humano como software, acceder a la red. Proporciona las interfaces de usuario y el soporte para servicios de información distribuida (transferencia de archivos, gestión de datos compartidos, etc.). Algunos de los servicios provistos por este nivel incluyen:

- Terminal virtual de red. Una máquina virtual de red es una versión de un terminal físico y permite al usuario acceder a una máquina remota. Para hacerlo la aplicación crea una emulación software de un terminal en la máquina remota, la computadora del usuario habla al terminal software que, a su vez, habla al host y viceversa.
- Transferencia, acceso y gestión de archivos (FTAM). Esta aplicación permite al usuario acceder a archivos en una computadora remota, recupera archivos y gestiona o controla los archivos en una computadora remota.
- Servicios de correo. Esta aplicación proporciona las bases para el envío y almacenamiento del correo electrónico.
- Servicios de directorios. Esta aplicación proporciona al acceso a bases de datos distribuidas que contienen información global sobre los distintos objetos y servicios.

1.2 Modelo TCP IP (Transmission Control Protocol / Internet Protocol)

El modelo TCP/IP fue desarrollado en la década de los 70 por Vinton Cerf y Robert E. Kahn. Fue implantado en la red ARPANET (Advanced Research Projects Agency Network), la primera red de área amplia, desarrollada por encargo de DARPA (Defense Advanced Research Projects Agency), una agencia del Departamento de Defensa de los Estados Unidos, y predecesora de la actual red Internet[2].

El modelo TCP/IP es una de las dos variantes principales disponibles en la red Internet. Es la asociación del protocolo Internet (IP) a un protocolo de control de transporte, que dirige un intercambio de datos entre dos máquinas en lo que podríamos llamar una "conversación" entre estas.

El modelo TCP/IP se impuso sobre el modelo OSI al simplificar el esquema de 7 capas en 4, también por que los protocolos TCP/IP son los estándares en torno a los cuales se desarrolló el internet, al mismo tiempo el modelo OSI fue propuesto como una aproximación teórica y también como una primera fase en la evolución de las redes, dejándolo como un modelo de referencia. Por tanto el modelo OSI es fácil de entender y es un marco conceptual que puede ser referenciado para comprender el funcionamiento de los dispositivos en la red, pero el modelo TCP/IP es el que realmente se usa.

Capa 1.- Nivel físico

Esta capa es asimilable a la capa 2 (enlace de datos) y a la capa 1 (física) del modelo OSI.

Esta capa engloba realmente las funciones de la de la capa física y la capa de enlace del modelo OSI. El modelo TCP/IP no dice gran cosa con respecto a ella, salvo que debe ser capaz de conectar el host a la red por medio de algún protocolo que permita enviar paquetes IP.

---

[2] Prácticas de redes, Varios autores (VV. AA.), Francisco Ortiz Zamora

Capa 2.- Nivel de red

Esta capa es asimilable a la capa 3 (red) del modelo OSI.

Es el „corazón" de la red, se encarga de encaminar los paquetes de la forma más conveniente para que lleguen a su destino, y de evitar que se produzcan situaciones de congestión en los nodos intermedios. Debido a los requisitos de robustez impuestos en el diseño, la capa de red da únicamente un servicio de conmutación de paquetes no orientado a conexión. Los paquetes pueden llegar desordenados a su destino, en cuyo caso es responsabilidad de las capas superiores en el nodo receptor la reordenación para que sean presentados al usuario de forma adecuada.

A diferencia de lo que ocurre en el modelo OSI, donde los protocolos para nada intervienen en la descripción del modelo, la capa de red define aquí un formato de paquete y un protocolo, llamado IP (Internet Protocol), que se considera el formato „oficial" de la arquitectura, siendo el más popular de todos.

Capa 3.- Nivel de transporte

Esta capa es asimilable a la capa 4 (transporte) del modelo OSI.

Desarrolla la misma función que la cuarta capa del modelo OSI, consiste en permitir la comunicación extremo a extremo (host a host) en la red. Aquí se definen dos protocolos:

- TCP (Transmission Control Protocol) ofrece un servicio fiable, con lo que los paquetes (aquí llamados segmentos) llegan ordenados y sin errores. También se ocupa del control de flujo extremo a extremo, para evitar que por ejemplo un host rápido sature a un receptor más lento.
- UDP (User Datagram Protocol) da un servicio no orientado a conexión y no fiable. UDP no realiza control de errores ni de flujo. Una aplicación típica donde se utiliza UDP es la transmisión de voz y video en tiempo real; aquí el retardo que introduciría el

control de errores produciría más daño que beneficio: es preferible perder algún paquete que retrasmitirlo fuera de tiempo.

Capa 4.- Nivel de aplicación

Esta capa es asimilable a las capas 5 (sesión), 6 (presentación) y 7 (aplicación) del modelo OSI.

Combina las funciones de las capas de sesión, presentación y aplicación del modelo OSI. La experiencia ha demostrado que las capas de sesión y presentación son de poca utilidad, debido a su escaso contenido, por lo que la aproximación adoptada por el modelo TCP/IP parece más acertada.

La capa de aplicación contiene todos los protocolos de alto nivel que se utilizan para ofrecer servicios al usuario. Entre éstos podemos mencionar los siguientes:

- Terminal virtual (TelNet).
- Transferencia de ficheros (FTP).
- Correo electrónico (SMTP).
- Servidor de nombres (DNS).
- Servicio de news (NNTP).
- Web (HTTP).

1.3 Tipos de Redes

Las redes reciben diversas clasificaciones para diferenciarlas unas de otras, la más reconocida es aquella que las distingue de acuerdo a su alcance, de esta manera los tipos de redes son:

- Red de área personal o PAN (*Personal Area Network*). Es una red conformada por una pequeña cantidad de equipos, establecidos a una corta distancia uno de otro,

permitiendo al usuario establecer una comunicación con estos dispositivos de manera rápida y eficaz.

- Red de área local, o LAN (Local Area Network). Es una red que conecta equipos en un área geográfica limitada (no mayor a 3 kilómetros), tal como una oficina o edificio. Su aplicación más extendida es la interconexión de computadoras personales y estaciones de trabajo en oficinas, fábricas o escuelas.
- Red de área de campus, o CAN (Campus Area Network). Es una red formada por varias redes LAN dispersadas geográficamente dentro de un campus (universitario, oficinas de gobierno, maquilas o industrias) pertenecientes a una misma entidad en una área delimitada en kilómetros. Por lo tanto, una red de área de campus es más grande que una red de área local, pero más pequeña que una red de área amplia.
- Red de área metropolitana, o MAN (Metropolitan Area Network). Es una red que comprende una ubicación geográfica determinada por ejemplo ciudad, municipio, y su distancia de cobertura es mayor de 4Kmts. Son redes con dos buses unidireccionales, cada uno de ellos es independiente del otro en cuanto a la transferencia de datos. Es básicamente una gran versión de LAN y usa una tecnología similar.
- Red de área amplia, o WAN (Wide Area Network). Es una re que conecta múltiples LAN entre sí a través de grandes distancias geográficas, que pueden llegar a extenderse hacia un país, un continente o el mundo entero. La conexión es realizada a través de fibra óptica, cables interoceánicos o satélites.

1.4 Topologías

La topología de red es la representación geométrica de la relación entre todos los enlaces y los dispositivos que los enlazan entre sí (habitualmente denominados nodos).
Actualmente existen al menos cinco posibles topologías de red básicas: bus, anillo, estrella, malla y árbol.

TOPOLOGÍA DE BUS

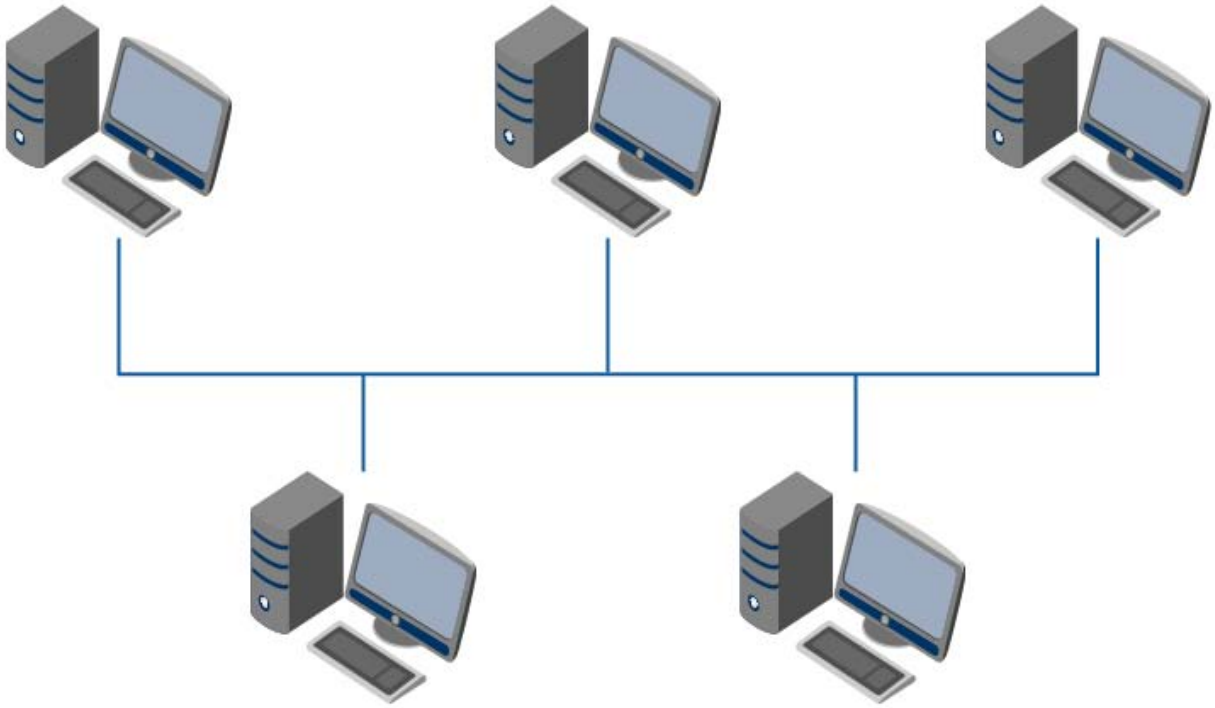

Figura 1: Topología en bus.

Una topología de bus es multipunto. Un cable largo actúa como una red troncal que conecta todos los dispositivos en la red.

Los nodos se conectan al bus mediante cables de conexión (latiguillos) y sondas. Un cable de conexión es el que va desde el dispositivo al cable principal. Una sonda es un conector que, o bien se conecta al cable principal, o se pincha en el cable para crear un contacto con el núcleo metálico.

Entre las ventajas de la topología de bus se incluye la sencillez de instalación. El cable troncal puede tenderse por el camino más eficiente y, después, los nodos se pueden conectar al mismo mediante líneas de conexión de longitud variable. De esta forma se puede conseguir que un bus use menos cable que una malla, una estrella o una topología en árbol.

TOPOLOGÍA EN ANILLO

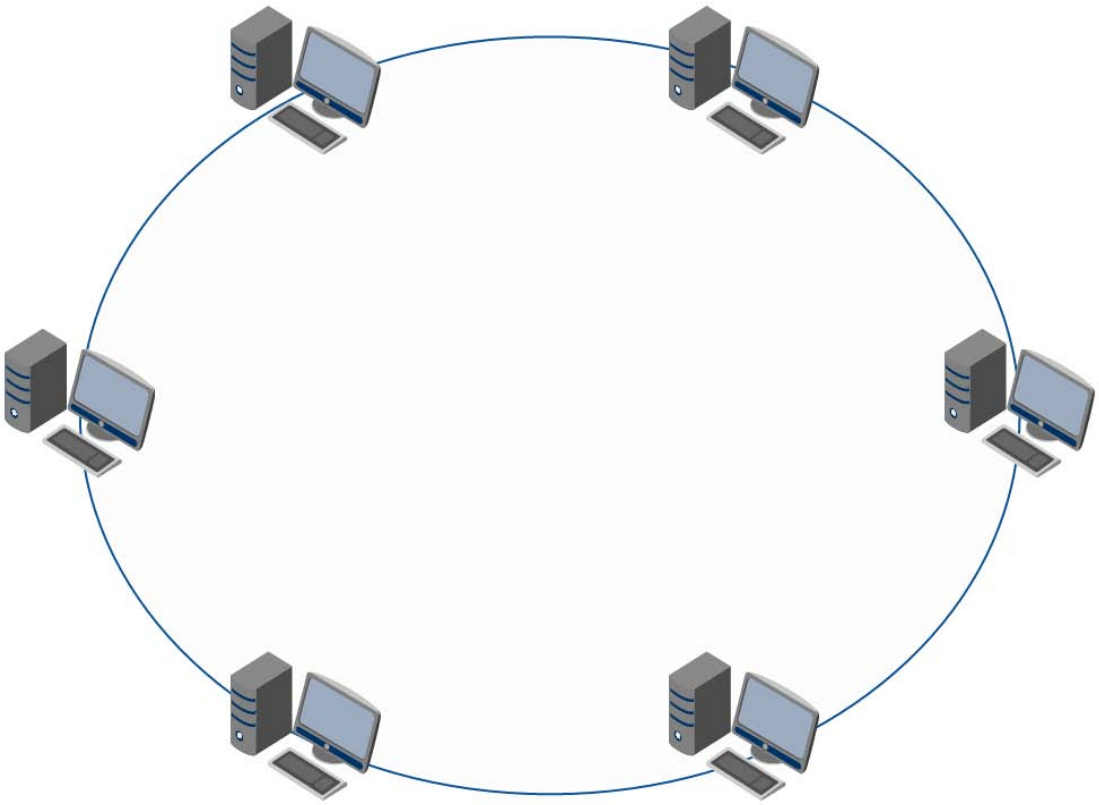

Figura 2: Topología en anillo.

En una topología en anillo cada dispositivo tiene una línea de conexión dedicada y punto a punto solamente con los dos dispositivos que están a sus lados. La señal pasa a lo largo del anillo en una dirección, o de dispositivo a dispositivo, hasta que alcanza su destino. Cada dispositivo del anillo incorpora un repetidor.

Un anillo es relativamente fácil de instalar y reconfigurar. Cada dispositivo está enlazado solamente a sus vecinos inmediatos (bien físicos o lógicos). Para añadir o quitar dispositivos, solamente hay que mover dos conexiones.

Las únicas restricciones están relacionadas con aspectos del medio físico y el tráfico (máxima longitud del anillo y número de dispositivos). Además, los fallos se pueden aislar de forma sencilla. Generalmente, en un anillo hay una señal en circulación continuamente.

TOPOLOGÍA EN ESTRELLA

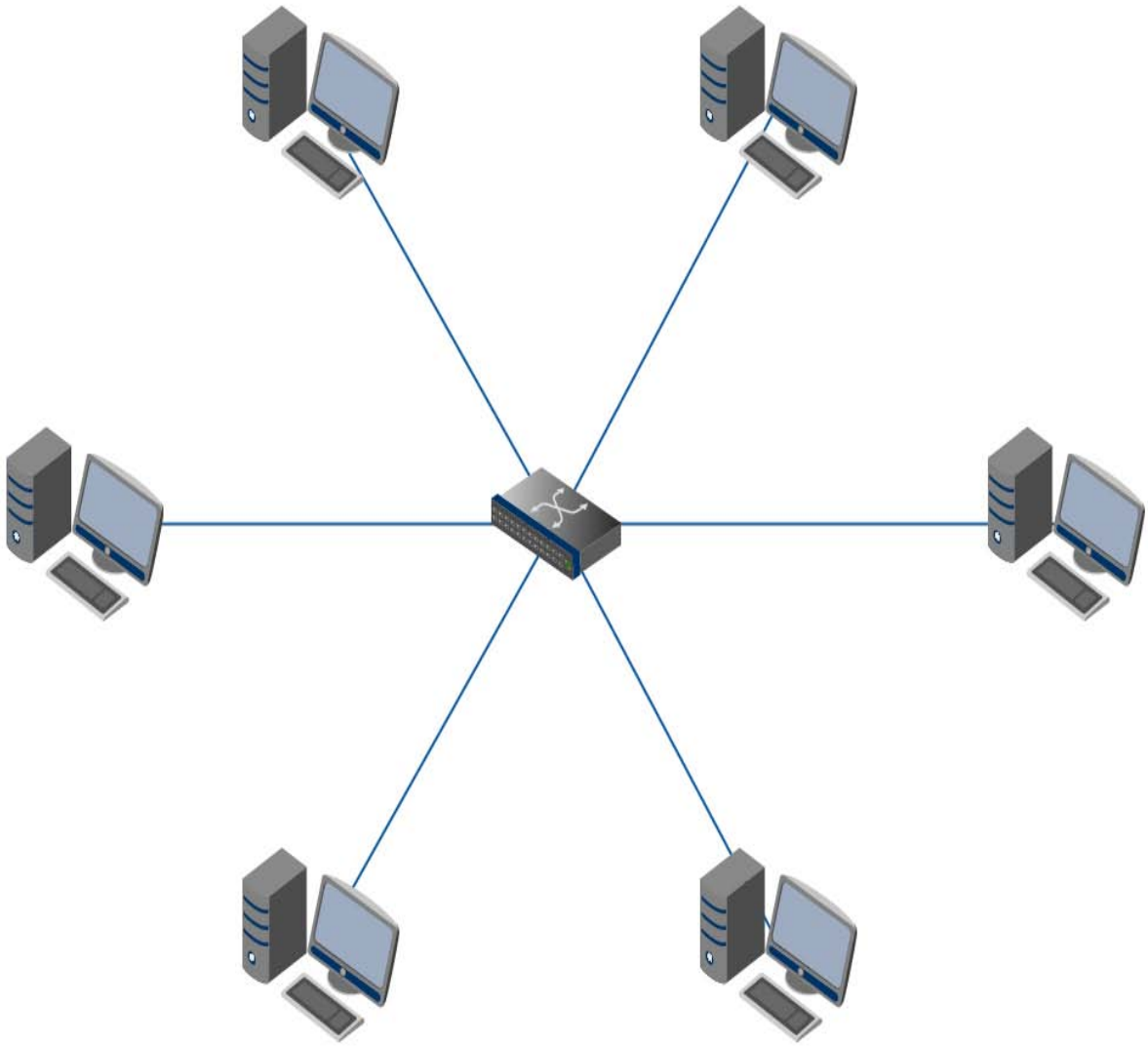

Figura 3: Topología en estrella.

En la topología en estrella cada dispositivo solamente tiene un enlace punto a punto dedicado con el controlador central, habitualmente llamado concentrador. Los dispositivos no están directamente enlazados entre sí.

El controlador actúa como un intercambiador: si un dispositivo quiere enviar datos a otro, envía los datos al controlador, que los retransmite al dispositivo final.

TOPOLOGÍA EN MALLA

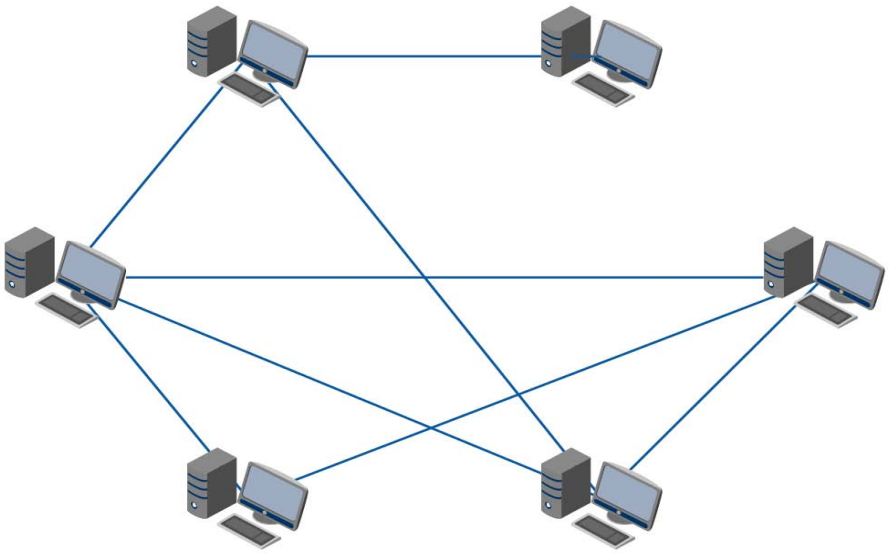

Figura 4: Topología en malla.

En una topología en malla, cada dispositivo tiene un enlace punto a punto y dedicado con cualquier otro dispositivo. El término dedicado significa que el enlace conduce el tráfico únicamente entre los dos dispositivos que conecta.

Por tanto, una red en malla completamente conectada necesita n(n-1)/2 canales físicos para enlazar n dispositivos. Para acomodar tantos enlaces, cada dispositivo de la red debe tener sus puertos de entrada/salida (E/S).

Una malla ofrece varias ventajas sobre otras topologías de red. En primer lugar, el uso de los enlaces dedicados garantiza que cada conexión sólo debe transportar la carga de datos propia de los dispositivos conectados, eliminando el problema que surge cuando los enlaces son compartidos por varios dispositivos. En segundo lugar, una topología en malla es robusta. Si un enlace falla, no inhabilita todo el sistema.

Otra ventaja es la privacidad o la seguridad. Cuando un mensaje viaja a través de una línea dedicada, solamente lo ve el receptor adecuado. Las fronteras físicas evitan que otros usuarios puedan tener acceso a los mensajes.

TOPOLOGÍA EN ÁRBOL

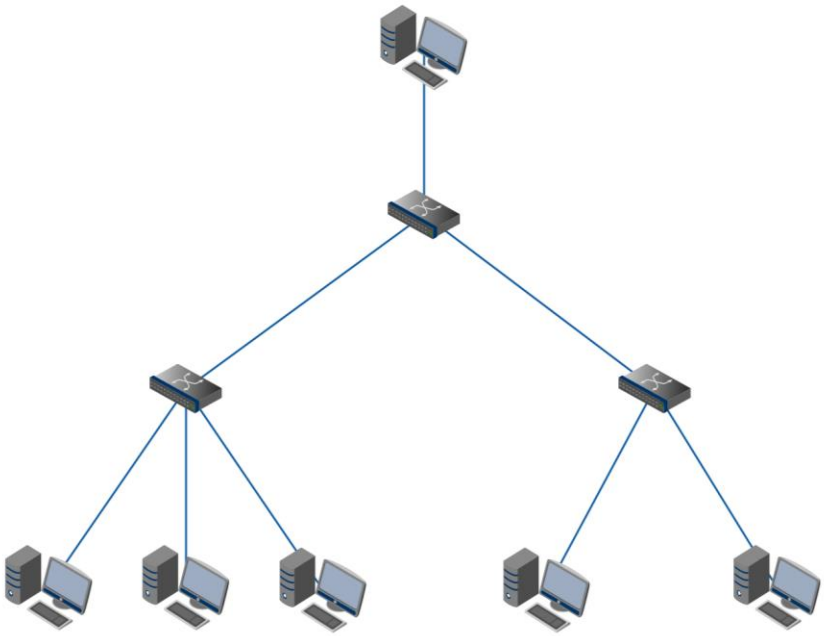

Figura 5: Topología en árbol.

La topología en árbol es una variante de la de estrella. Como en la estrella, los nodos del árbol están conectados a un concentrador central que controla el tráfico de la red. Sin embargo, no todos los dispositivos se conectan directamente al concentrador central. La mayoría de los dispositivos se conectan a un concentrador secundario que, a su vez, se conecta al concentrador central.

1.5 Equipo Activo y Equipo Pasivo

Las redes de computadoras están conformadas por equipos activos y pasivos, que se interconectan para su desarrollo y funcionamiento.

a) Los equipos activos son aquellos que se encargan de distribuir la información a través de la red y como estos están conectados a la energía eléctrica para su correcto funcionamiento, ayudan a amplificar la señal contrarrestando el deterioro de la misma.

- Hub. (capa1) Son utilizados para conectar dos o más segmentos de una red. A medida

que los segmentos exceden su longitud máxima, la calidad de la señal comienza a deteriorarse.

Los hubs proveen la amplificación de señal requerida para permitirle a un segmento extenderse a una distancia mayor. Un hub toma cualquier señal entrante y la repite a todos los restantes puertos de salida.

- Bridge. Es un dispositivo de interconexión de redes de ordenadores que opera en la capa 2 (nivel de enlace de datos) del modelo OSI. Este interconecta segmentos de red (o divide una red en segmentos) haciendo la transferencia de datos de una red hacia otra con base en la dirección física de destino de cada paquete.
- Gateway. (capa2) Consiste en una computadora u otro dispositivo que actúa como traductor entre dos sistemas que no utilizan los mismos protocolos de comunicaciones, formatos de estructura de datos, lenguajes y/o arquitecturas. Una pasarela no es como un puente, que simplemente transfiere la información entre dos sistemas sin realizar conversión. Un gateway modifica el empaquetamiento de la información o su sintaxis para acomodarse al sistema destino.
- Router. Un router opera en la capa 3 del modelo OSI y tiene más facilidades de software que un switch. Al funcionar en una capa mayor que la del switch, puede distinguir entre los diferentes protocolos de red, tales como IP, IPX, AppleTalk o DECnet. Los routers conectan redes tanto en las áreas locales como en las extensas, y cuando existen más de una ruta entre dos puntos finales de red, proporcionan control de tráfico y filtrado de funciones.
- Switch. Un switch es un dispositivo de proposito especial diseñado para resolver problemas de rendimiento en la red.

  En su presentación es muy parecido al hub, sólo difiere en su función lógica y en la adición de unos puertos para funciones adicionales.

  El switch puede acelerar la salida de paquetes, reducir tiempo de espera y bajar el costo por puerto. Opera en la capa 2 del modelo OSI y reenvía los paquetes en base a la dirección MAC.

b) Los equipos pasivos se utilizan para interconectar los enlaces de una red de datos su utilización se define en las normativas internacionales.

- Cable de par trenzado. Consta de ocho hilos de cobre aislados y entrelazados. Hay dos tipos de cables de par trenzado: cable de par trenzado sin apantallar el cable UTP es el tipo más conocido de cable de par trenzado y ha sido el cableado LAN más utilizado en los últimos años.
- Cable coaxial. Consta de un núcleo de hilo de cobre rodeado por un aislante, un apantallamiento de metal trenzado y una cubierta externa.
- Fibra óptica. La fibra óptica es un medio excelente para la transmisión de información por sus características: gran ancho de banda, baja atenuación de la señal que permite cubrir grandes distancias sin repetidores, integridad (proporción de errores baja), inmunidad a interferencias electromagnéticas, alta seguridad y larga duración, resistente a la corrosión y altas temperaturas. Sus mayores desventajas son su coste de producción superior al resto de los tipos de cable y su fragilidad durante el manejo en producción.
- Canaleta. Medio de protección y enrutamiento del cableado de red y cableado eléctrico.
- Patch Panel. Los llamados Patch Panel son utilizados en algún punto de una red informática donde todos los cables de red terminan. Se puede definir como paneles donde se ubican los puertos de una red, normalmente localizados en un bastidor o rack de telecomunicaciones. Todas las líneas de entrada y salida de los equipos tendrán su conexión a uno de estos paneles.
- Rack de comunicaciones. Es un gabinete necesario y recomendado para instalar el patch panel y los equipos activos proveedores de servicios.

1.6 Sistemas operativos para redes de computadoras

Un sistema operativo para red, o NOS (Network Operating Sistem) es el conjunto de programas y protocolos de comunicación que permite a las distintas computadoras de la red

comunicarse entre sí y compartir recursos de manera organizada, eficiente y transparente[3].

Los sistemas operativos de red pueden clasificarse por su estructura o por la forma de ofrecer sus servicios[4].

a) Por su estructura:

1. **Sistemas operativos de estructura monolítica**, es decir, están compuestos por un solo programa que contiene varias rutinas entrelazadas de forma que una de ellas puede comunicarse fácilmente con el resto.
2. **Sistemas operativos de estructura jerárquica.** Con el tiempo los sistemas operativos se van mejorando y se hizo necesario que el sistema operativo tuviese varias partes bien definidas del resto y con una interface propia. Con estas partes, capas o niveles se pretendía entre otras que cosas que las partes más importantes del sistema operativo estuviesen a salvo de intrusos.
3. **Sistemas operativos cliente-servidor.** Este tipo de sistemas operativos para redes es el que actualmente está en uso en la mayoría de las computadoras. Es un sistema operativo muy compatible, porque sirven para cualquier computadora y prácticamente para todos los programas. El sistema operativo cliente-servidor el usuario o cliente hace una petición al servidor correspondiente para tener acceso a un archivo o efectuar una actuación de entrada o salida sobre un dispositivo concreto.

b) Por la forma de ofrecer sus servicios:

1. **Sistemas operativos de red.** Son aquellos sistemas operativos capaces de relacionarse eficientemente con sistemas operativos instalados en otras computadoras transmitiendo o intercambiando información, archivos, ejecutando comandos remotos, etc.

---

[3] Tecnologías y redes de transmisión de datos, Enrique Herrera Pérez

[4] http://www.larevistainformatica.com/clasificacion-sistemas-operativos-redes.html

2. **Sistemas operativos distribuidos.** Este tipo de sistema operativo de red lleva a cabo todos los servicios que realizaba el sistema operativo de red, pero además consigue compartir más recursos como impresoras, memorias, unidades centrales de proceso (CPU), discos duros, etc. Y el usuario no necesita saber la ubicación del recurso, ni ejecutar comandos, sino que los conoce por nombres y los usa como si fuesen locales o propios de su computadora.

1.7 Modelo Cliente-Servidor

La arquitectura cliente-servidor es un modelo para el desarrollo de sistemas de información, en el que las transacciones se dividen en procesos independientes que cooperan entre sí para intercambiar información, servicios o recursos. Se denomina cliente al proceso que inicia el diálogo o solicita los recursos y servidor, al proceso que responde a las solicitudes. Es el modelo de interacción más común entre aplicaciones en una red.

- Cliente. Conjunto de software y hardware que invoca los servicios de uno o varios servidores. Los clientes interactúan con el usuario, usualmente en forma gráfica. Frecuentemente se comunican con procesos auxiliares que se encargan de establecer conexión con el servidor, enviar el pedido, recibir la respuesta, manejar las fallas y realizar actividades de sincronización y de seguridad.
- Servidor. Conjunto de hardware y software que responde a los requerimientos de un cliente. Los servidores proporcionan un servicio al cliente y devuelven los resultados. En algunos casos existen procesos auxiliares que se encargan de recibir las solicitudes del cliente, verificar la protección, activar un proceso-servidor para satisfacer el pedido, recibir su respuesta y enviarla al cliente. Además, deben manejar los interbloqueos, la recuperación ante fallas, y otros aspectos afines. Por las razones anteriores, la plataforma computacional asociada con los servidores es más poderosa que la de los clientes. Por esta razón se utilizan PCs poderosas, estaciones de trabajo, minicomputadores o sistemas grandes. Además deben manejar servicios como administración de la red, mensajes, control y administración de la entrada al sistema ("login"), auditoría, recuperación y contabilidad.

Una infraestructura Cliente/Servidor consta de tres componentes esenciales, todos ellos de igual importancia y estrechamente ligados[5]:

- Plataforma Operativa. La plataforma deberá soportar todos los modelos de distribución Cliente/Servidor, todos los servicios de comunicación, y deberá utilizar, preferentemente, componentes estándar de la industria para los servicios de distribución. Los desarrollos propios deben coexistir con las aplicaciones estándar y su integración deberá ser imperceptible para el usuario. Igualmente, podrán acomodarse programas escritos utilizando diferentes tecnologías y herramientas.
- Entorno de Desarrollo de Aplicaciones. Debe elegirse después de la plataforma operativa. Un entorno de aplicación incremental, debe posibilitar la coexistencia de procesos cliente y servidor desarrollados con distintos lenguajes de programación y/o herramientas, así como utilizar distintas tecnologías (por ejemplo, lenguaje procedural, lenguaje orientado a objetos, multimedia), y que han sido puestas en explotación en distintos momentos del tiempo.
- Gestión de Sistemas. - Estas funciones aumentan considerablemente el costo de una solución, pero no se pueden evitar. Siempre deben adaptarse a las necesidades de la organización, y al decidir la plataforma operativa y el entorno de desarrollo, es decir, en las primeras fases de la definición de la solución, merece la pena considerar los aspectos siguientes:
    - ¿Qué necesitamos gestionar?
    - ¿Dónde estarán situados los procesadores y estaciones de trabajo?
    - ¿Cuántos tipos distintos se soportarán?
    - ¿Qué tipo de soporte es necesario y quién lo proporciona?

1.8 Servidores Dedicados y No dedicados

Servidor Dedicado

Es un equipo informático, construido y desarrollado para trabajar en condiciones de alto

[5] http://labredes.itcolima.edu.mx/fundamentosbd/sd_u1_6.htm

rendimiento que permite a los usuarios disponer de él las 24 horas del día 365 días del año[6].

Este equipo normalmente se encuentra en Datacenters refrigerados con sistemas de seguridad y de energía eléctrica redundante, así como conectividad nacional e internacional.

Con un servidor dedicado se puede ejecutar todo tipo de aplicaciones en una máquina con las características de hardware que más se ajusten a las necesidades de los usuarios.

Servidores de correos, bases de datos, servidores web, de juegos son algunos de los principales servicios que se pueden ofrecer en un servidor dedicado.

Algunas características de los servidores dedicados son las siguientes:

- Flexibilidad y personalización.- Los clientes pueden elegir el hardware y las especificaciones exactas del software del servidor.
- Recursos dedicados y resultados garantizados.- Los clientes no comparten los recursos del servidor con ninguna otra persona.
- Control total.- Los clientes tienen total acceso a sus servidores y pueden configurarlo según las características que requieran.

Servidor no Dedicado

Al igual que un servidor dedicado, un servidor no dedicado es un equipo informático, diseñado para trabajar en condiciones de alto rendimiento que permite disponer de él las 24 horas del día 365 días del año.

La diferencia radica en que los recursos de un servidor no dedicado se comparten con otros usuarios para servir también como estaciones de trabajo para usuarios locales.

---

[6] http://hostingexpress.com.mx/servidores-dedicados.html

1.9 Clasificación de servidores

De acuerdo a la función que se le quiera dar a un servidor, estos se pueden clasificar en:

- Servidores de Aplicaciones (Application Servers): Designados a veces como un tipo de middleware (software que conecta dos aplicaciones), los servidores de aplicaciones ocupan una gran parte del territorio entre los servidores de bases de datos y el usuario, y a menudo los conectan.
- Servidores de Audio/Video (Audio/Video Servers): Estos servidores añaden capacidades multimedia a los sitios web permitiéndoles mostrar contenido multimedia en forma de flujo continuo (streaming) desde el servidor.
- Servidores de Chat (Chat Servers): Permiten intercambiar información a una gran cantidad de usuarios ofreciendo la posibilidad de llevar a cabo discusiones en tiempo real.
- Servidores de Fax (Fax Servers): Un servidor de fax es una solución ideal para organizaciones que tratan de reducir el uso del teléfono pero necesitan enviar documentos por fax.
- Servidores FTP (FTP Servers): Uno de los servicios más antiguos de Internet, File Transfer Protocol permite mover uno o más archivos con seguridad entre distintos ordenadores proporcionando seguridad y organización de los archivos así como control de la transferencia.
- Servidores Groupware (Groupware Servers): Un servidor groupware es un software diseñado para permitir colaborar a los usuarios, sin importar la localización, vía Internet o vía Intranet corporativo y trabajar juntos en una atmósfera virtual.
- Servidores IRC (IRC Servers): Otra opción para usuarios que buscan la discusión en tiempo real, Internet Relay Chat consiste en varias redes de servidores separadas que permiten que los usuarios conecten el uno al otro vía una red IRC.
- Servidores de Listas (List Servers): Los servidores de listas ofrecen una manera mejor de manejar listas de correo electrónico, bien sean discusiones interactivas abiertas al público o listas unidireccionales de anuncios, boletines de noticias o publicidad.

- Servidores de Correo (Mail Servers): Casi tan ubicuos y cruciales como los servidores web, los servidores de correo mueven y almacenan el correo electrónico a través de las redes corporativas (vía LANs y WANs) y a través de Internet.
- Servidores de Noticias (News Servers): Los servidores de noticias actúan como fuente de distribución y entrega para los millares de grupos de noticias públicos actualmente accesibles a través de la red de noticias USENET.
- Servidores Proxy (Proxy Servers): Los servidores proxy se sitúan entre un programa del cliente (típicamente un navegador) y un servidor externo (típicamente otro servidor web) para filtrar peticiones, mejorar el funcionamiento y compartir conexiones.
- Servidores Telnet (Telnet Servers): Un servidor telnet permite a los usuarios entrar en un ordenador huésped y realizar tareas como si estuviera trabajando directamente en ese ordenador.
- Servidores Web (Web Servers): Básicamente, un servidor web sirve contenido estático a un navegador, carga un archivo y lo sirve a través de la red al navegador de un usuario. Este intercambio es mediado por el navegador y el servidor que hablan el uno con el otro mediante HTTP.
- Servidores de impresiones (Print Server): controla una o más impresoras y acepta trabajos de impresión de otros clientes de la red, poniendo en cola los trabajos de impresión (aunque también puede cambiar la prioridad de las diferentes impresiones), y realizando la mayoría o todas las otras funciones que en un sitio de trabajo se realizaría para lograr una tarea de impresión si la impresora fuera conectada directamente con el puerto de impresora del sitio de trabajo.
- Servidores de base de datos (Database Server): provee servicios de base de datos a otros programas u otras computadoras, como es definido por el modelo cliente-servidor. También puede hacer referencia a aquellas computadoras (servidores) dedicadas a ejecutar esos programas, prestando el servicio.

1.10 Servidor Web

Un servidor web o servidor HTTP es un programa informático que procesa una aplicación del

lado del servidor realizando conexiones bidireccionales y/o unidireccionales y síncronas o asíncronas con el cliente generando o cediendo una respuesta en cualquier lenguaje o aplicación del lado del cliente.

Generalmente se utiliza el protocolo HTTP o el protocolo HTTPS (la versión cifrada y autenticada) para estas comunicaciones, perteneciente a la capa de aplicación del modelo OSI.

Un servidor web básico cuenta con un esquema de funcionamiento muy simple, basado en ejecutar infinitamente el siguiente bucle:

1. Espera peticiones en el puerto TCP indicado (el estándar por defecto para HTTP es el 80 y 443 para HTTPS).
2. Recibe una petición.
3. Busca el recurso.
4. Envía el recurso utilizando la misma conexión por la que recibió petición.
5. Vuelve al segundo punto.

Un servidor web que siga el esquema anterior cumplirá todos los requisitos básicos de los servidores HTTP, aunque sólo podrá servir ficheros estáticos.

A partir del anterior esquema se han diseñado y desarrollado todos los servidores de HTTP que existen, variando sólo el tipo de peticiones (páginas estáticas, CGIs, Servlets, etc.) que pueden atender, en función de que sean o no sean multi-proceso o multi-hilados, etc.

a) Servicio de ficheros estáticos

Todos los servidores web deben incluir, al menos, la capacidad para servir los ficheros estáticos que se hallen en alguna parte del disco. Un requisito básico es la capacidad de especificar qué parte del disco se servirá. No resulta recomendable que el programa servidor obligue a usar un directorio concreto, aunque sí puede tener uno por defecto.

La mayoría de servidores web permiten añadir otros directorios o subdirectorios para servir, especificando en qué punto del "sistema de ficheros" virtual del servidor se localizarán los recursos.

Algunos servidores web permiten también especificar directivas de seguridad (quién puede acceder a los recursos), mientras que otros hacen posible la especificación de los ficheros que se deben considerar como índice del directorio.

b) Contenido dinámico

Uno de los aspectos fundamentales del servidor web elegido es el nivel de soporte que ofrece para servir contenido dinámico. Puesto que la mayor parte del contenido web que se sirve no viene de páginas estáticas, sino que se genera de forma dinámica, y esta tendencia se mueve claramente al alza, el soporte para contenido de tipo dinámico que ofrece un servidor web es uno de los puntos críticos en la elección.

La mayor parte de los servidores web ofrecen soporte para CGI (se debe recordar que los CGI son el método más antiguo y sencillo para generar contenido dinámico). Otros muchos ofrecen soporte para algunos lenguajes de programación (normalmente lenguajes interpretados) como PHP, JSP, ASP, etc. Es muy recomendable que el servidor web proporcione soporte para algunos de estos lenguajes, especialmente PHP, sin tener en cuenta JSP, que normalmente requerirá un software externo para funcionar (como un contenedor de Servlets). La oferta es muy amplia, pero antes de elegir un lenguaje de programación de servidor se debe plantear si se desea un lenguaje muy estándar para que la aplicación no dependa de un servidor web o una arquitectura concreta o si, al contrario, la portabilidad no es prioritaria y sí lo es alguna otra prestación concreta que pueda ofrecer algún lenguaje de programación concreto.

El término “servidor web” también se emplea para referirse al equipo de cómputo que ejecuta el programa.

1.11 Servidor de Base de Datos

Un servidor de bases de datos se utiliza para la ejecución de gestores de BD y de la posibilidad a que varios usuarios, situados en lugares diferentes, hagan operaciones sobre ellas al mismo tiempo.

Los servidores de bases de datos surgen en la década del 80 con motivo de la necesidad de las empresas de manejar grandes y complejos volúmenes de datos, al tiempo que requieren compartir la información con un conjunto de clientes (que pueden ser tanto aplicaciones como usuarios) de una manera segura y debe proporcionar servicios de forma global y, en la medida de lo posible, independientemente de la plataforma.

En todo sistema abierto, debe proporcionarse un potente mecanismo de seguridad que garantice que ningún intruso pueda acceder o corromper la integridad del sistema, en servidores de bases de datos hablaremos de la seguridad a 4 niveles básicos:

- Seguridad de acceso al sistema: se implementa de dos maneras posibles: a nivel de sistema operativo, en cuyo caso el SGBD se apoya en la seguridad de entrada al sistema operativo para comprobar la validez del acceso a los datos almacenados; o bien lo que se llama modo mixto, en el cual la seguridad de entrada a la información la llevará a cabo el propio servidor de datos a partir de la definición de cuentas de usuario al servidor (su denominación de mixta proviene de la capacidad de los sistemas de incluir como cuentas de acceso a las cuentas propias del sistema operativo, lo que facilita la transición de las cuentas de seguridad). La segunda es de gran ayuda cuando los clientes que acceden al sistema provienen de sistemas operativos con poca (o ninguna) seguridad o de aplicaciones instaladas que necesiten acceder a los volúmenes de información del sistema. En ambos casos, en los sistemas se contará con roles o papeles con los que contará el usuario al entrar al sistema para la realización de determinadas operaciones de cara al sistema.
- Seguridad a nivel de objetos de datos: es el acceso a nivel de creación y administración de objetos de datos: tablas, vistas, índices, relaciones, reglas...etc. Es decir, las

responsabilidades y acciones que puede hacer el usuario en el esquema de la base de datos (el esqueleto a partir del cual el sistema definirá cómo se debe almacenar y relacionar la información). Este podrá especificar de nuevo roles a los usuarios, indicando quién podrá crear, modificar o eliminar cualquier objeto de datos (con lo que se permite establecer una política de delegación de responsabilidades).

- Seguridad a nivel de datos: accede a la información para su consulta, actualización, inserción o borrado y las características de los diversos motores, los que determinarán hasta qué grado de seguridad se llega en este apartado (desde la protección de las columnas de una tabla hasta la tabla misma, creación de vistas...etc.).
- Seguridad en cuanto a protección de los almacenamientos físicos de los datos: es la seguridad a nivel de sistema operativo de los archivos de datos del sistema, y las políticas de copia de seguridad y restauración de los datos (tanto con herramientas del sistema operativo como las proporcionadas por el propio servidor de datos) junto con sus posibles aproximaciones (total, incremental y diferencial), además de los soportes hardware compatibles de almacenamiento masivo empleados como destino de las copias.

1.12 Gestión de la Seguridad

En la actualidad el flujo continuo de datos, se ha convertido en un factor crítico de la era digital.

Garantizar la integridad, disponibilidad y confidencialidad de la información, es la principal función de la seguridad dentro de una organización.

INFOSEC Glossary 2000 define la seguridad informática como “son medidas y controles que aseguran la confidencialidad, integridad y disponibilidad de los activos de los sistemas de información, incluyendo hardware, software, firmware y aquella información que procesan, almacenan y comunican”.

El propósito general de la seguridad de información es controlar y reducir los riesgos posibles

ocasionados por intrusos a los recursos de una organización, asegurando el uso correcto de la información y de todo recurso informático.

Amenazas y ataques

Cualquier evento que pueda afectar o interrumpir con la continuidad del flujo de la información en un sistema informático, es considerado como una amenaza a la información.

Las amenazas se pueden dividir en:

- Accidentes: Son ocasionados por sucesos no provocados, por ejemplo:
    - Desastres naturales: terremotos, inundaciones, tormentas, etc.
    - Fallas en hardware: defectos de fabricación, desgaste por uso, fin de vida útil o afectaciones por estática o temperatura.
    - Fallas de software: código de un sistema mal depurado, actualización de software del sistema operativo, incompatibilidad de versiones de software.
    - Fallas de comunicación: mala calidad o degradación de los medios de transmisión de la información.
    - Fallas de almacenamiento: defectos de fábrica, daños causados por el uso o fin de vida útil de medios de almacenamiento que pueden impedir la recuperación de la información.
- Errores: Son ocasionados por la falta de experiencia o información del personal que interactúa con los sistemas de información, por ejemplo:
    - Permanencia incontrolada de información en equipos de cómputo: información que se queda almacenada en cookies, memoria RAM, cache, archivos temporales o bases de datos.
    - Fraudes, cadenas de correo y spam: todo aquel correo en el que se recibe información falsa u obsoleta o redireccionamiento a páginas falsas.
    - Prestamos de usuarios y contraseñas: es una mala práctica y muy frecuente por parte de los usuarios el compartir las claves de acceso de un equipo.

Lo ataques son ocasionados por personas que pueden o no pertenecer a una organización.
Los ataques más comunes son:

- Reconocimiento de sistemas: su objetivo es lograr obtener información sobre las organizaciones, sus redes y sistemas informáticos.
- Detección de vulnerabilidades: identificar vulnerabilidades en los sistemas, para después desarrollar herramientas que permitan un acceso fácil.
- Robo de información mediante la intercepción de mensajes: interceptar información que es enviada por un medio de información no protegido.
- Modificación de contenido: modificación de la información de un mensaje interceptado para después ser reenviado.
- Análisis de tráfico: observar y analizar los datos transmitidos a través de las redes.
- Suplantación de identidad: simular ser un equipo de la red para lograr acceso a otros equipos o el robo de la información.
- Modificación de tráfico y de tablas de enrutamiento: desviar paquetes de tráfico de una red a equipos intermedios para facilitar el robo de la información.
- Ataques contra sistemas de encriptación: tienen como propósito conseguir las claves utilizadas para encriptar mensajes o documentos.
- Introducción de código malicioso (malware): es cualquier documento, programa o correo capaz de causar un daño (virus, gusanos, troyanos, bombas lógicas, spyware, etc.).
- Keyloggers: programas que registran cada dato tecleado en la computadora.
- Denegación de servicios (DoS DDoS): su finalidad es colapsar los recursos de una red para impedir que ofrezcan sus servicios.

Identificación de vulnerabilidades y amenazas

Algunas recomendaciones en seguridad de la información son:

- Hacer un inventario de recursos informáticos a proteger.

- Identificar vulnerabilidades y amenazas.
- Implementar políticas y procedimientos basados en algún estándar.
- Realizar auditorías de seguridad para la evaluación del funcionamiento de políticas y procedimientos.

Inventario de recursos informáticos

Un inventario de recursos informáticos es una lista detallada de los activos tangibles e intangibles de una organización.

- Activos tangibles: Los activos tangibles son todas las computadoras personales, equipos portátiles (Laptops), servidores, impresoras, medios de almacenamiento, dispositivos de red, teléfonos, equipos eléctricos, entre otros.
- Activos intangibles: Los activos intangibles son el software y las aplicaciones usadas por la organización, patentes, información de clientes y de trabajadores, entre otros.

Identificación de vulnerabilidades y amenazas

Una prueba de penetración (también llamado Pentest) es una técnica de evaluación de la seguridad de los sistemas de información de una organización, imitando las acciones realizadas por un atacante.

Su objetivo es realizar un análisis activo para identificar las vulnerabilidades, errores de configuración de sistemas y de dispositivos, determinando el impacto que pueda tener.

El Instituto para la Seguridad y las Metodologías Abiertas (ISECOM) identifica las pruebas de seguridad de Internet a los diferentes tipos de sistemas, basados en tiempo y costo:

- Búsqueda de vulnerabilidades: representa a las verificaciones automáticas de un sistema o red.

- Escaneo de seguridad: incluye las comprobaciones manuales y el análisis de falsos positivos e identificación de puntos de acceso débiles en la red.
- Test de intrusión: agrupa a la identificación de vulnerabilidades de los programas y aplicaciones.
- Evaluación de riesgo: es el análisis de seguridad efectuado a través de entrevistas e investigación que incluye justificación de negocios, legales y específicas del ramo de la que pertenece la organización a evaluar.
- Auditoría de seguridad: es la supervisión manual de sistemas operativos y aplicaciones dentro de la red, con acceso autorizado.
- Hacking ético: incluye las pruebas de intrusión para encontrar vulnerabilidades a fin de lograr acceso no autorizado a los recursos.

Políticas y procedimientos basados en un estándar

Un estándar es un conjunto de documentos, que indican como implantar, controlar, analizar y mejorar la administración de la seguridad en una organización.

Un programa de seguridad basa su funcionamiento en el correcto diseño de políticas, pero no indican las actividades que se deben seguir para lograrlo, por esto surge la necesidad de utilizar estándares de seguridad, los cuales especifican:

- Lo que se debe de hacer.
- Lo que se necesita controlar.
- La adaptación de los controles necesarios, de acuerdo al entorno de la organización para la protección de la información.

Después de identificar las vulnerabilidades y amenazas, es necesario definir las políticas y realizar las acciones necesarias para implementarlas.

Las políticas de seguridad determinan las necesidades y requerimientos de resguardo en una organización, comunican de una manera formal la conducta del personal describiendo lo que se desea resguardar y por qué hacerlo, así mismo debe estar redactada de una forma sencilla y clara para facilitar su entendimiento y comprensión.

El éxito de la implantación de las políticas de seguridad, dependerá esencialmente de realizar un programa de sensibilización, explicando la importancia del uso de las políticas para mejorar las funciones de cada trabajador, eliminando la percepción de imponer medidas burocráticas para vigilar su trabajo.

La academia latinoamericana de seguridad informática recomienda tener en cuenta las siguientes sugerencias para la implementación de las políticas con la finalidad de disminuir las vulnerabilidades y amenazas:

- Usar un Firewall.
- Contar con un control de acceso perimetral.
- Hacer uso de sistemas de vigilancia.
- Hacer uso de un sistema de detección de intrusos.
- Hacer una configuración personalizada de dispositivos, recursos y equipos de red.
- Capacitación de personal.
- Contar con un plan de continuidad.
- Contar con un plan de recuperación de desastres.
- Clasificar los activos.
- Hacer uso de VPN‟s.
- Hacer uso de accesos remotos seguros.
- Monitorear los recursos de la red.
- Administrar el sistema de seguridad.

Auditorias de seguridad

Una auditoria de seguridad consiste en efectuar una evaluación a la infraestructura tecnológica de la empresa a través de un análisis detallado a nivel técnico sobre servidores, computadoras de trabajo y personales, red, firewall, contraseñas, accesos, plan de contingencia, antivirus, actualizaciones, etc.

Para su realización se usa todo tipo de software y herramientas para detectar los problemas relacionados con la seguridad existente.

Los beneficios que se obtienen de esta área son:

- Tener conciencia del total de información de la organización.
- Identificar activos con mayor necesidad de protección.
- Medir niveles de efectividad de policitas y procedimientos de seguridad.
- Reportar fallas y vulnerabilidades.
- Obtener un reporte de efectividad de la seguridad de la información.
- Actualizar las medidas de seguridad de acuerdo a los resultados obtenidos.

CAPÍTULO II TECNOLOGÍAS DE CLUSTER

Actualmente la Tecnología de Clusters ya no se refiere a un solo tipo de arquitectura, ni su uso está limitado al campo de la investigación, la evolución del cómputo y las necesidades que surgen en cuanto a la disponibilidad de la información, han hecho que esta tecnología se aproveche como una buena solución para otros fines y no solo para lograr un alto desempeño en cálculos científicos.

El termino cluster se refiere a un grupo de computadoras trabajando juntas como un solo sistema con un fin específico este puede ser para lograr alto rendimiento, alta disponibilidad o balanceo de carga según el problema que se desea resolver.

La principal característica de las Tecnologías de Clusters es que el grupo de computadoras que lo componen es visto y administrado como una sola entidad a través de un canal de comunicación, está construido para tolerar fallas en cualquiera de sus componentes así como para soportar la adición o remoción de nodos de forma transparente para el usuario.

2.1 Componentes

Los componentes de una arquitectura básica de clusters se pueden resumir en nodos, canal de comunicación y software de control.

Los nodos son los equipos que integraran al cluster estos pueden ser de arquitectura y características idénticas o distintas dependiendo del uso que se les dará, se encuentran interconectados entre si y proporcionaran diferentes recursos.

Se puede dividir o clasificar a los nodos dentro de un cluster según su función; los nodos especializados por ejemplo realizan alguna labor en específico dentro del cluster, generalmente se encargan de administrar los servicios de los nodos que se encuentran detrás de ellos dentro de la topología, también pueden existir nodos de respaldo o secundarios, los cuales están a la

espera de que ocurra algún evento en específico para comenzar a prestar sus servicios. Las clasificaciones y funciones de los nodos dependen directamente del tipo de cluster que se esté construyendo.

Canal de comunicación. Para que un cluster pueda comunicarse con cada uno de sus nodos, es necesario que estos estén interconectados, generalmente un switch que conecte a todos los nodos entre sí basta para una configuración básica. Sin embargo seleccionar la tecnología de intercomunicación depende de varios factores, como la compatibilidad del hardware y sistema operativo del cluster, precio y rendimiento.

El rendimiento de la conexión se mide por medio de 2 métricas: ancho de banda y latencia. Ancho de banda es la cantidad de datos que se pueden transmitir a través del hardware de conexión en un periodo de tiempo, mientras que latencia es el tiempo de preparación y transmisión de datos por parte del nodo fuente al nodo destino[7].

Software de control. Son servicios que corren sobre el sistema operativo en uno o varios de los nodos que integran al cluster, estos determinan las acciones a realizar por los equipos dentro de la infraestructura, su función depende del tipo de cluster que se está utilizando, por ejemplo, en un cluster de alta disponibilidad, el software de control se encarga de comprobar el estado de algún equipo, y en caso de detectar alguna falla, comienza a realizar determinadas acciones para que ese servicio no se vea interrumpido.

[7]Cluster Computing: High-Performance, High-Availability, and High-Throughput Processing on a Network of Computers, Chee Shin Yeo

2.2 Cluster de alto rendimiento (HP)

Este tipo de clusters están diseñados para brindar una mayor capacidad en el procesamiento de datos. Se utilizan principalmente en la resolución de problemas que requieren grandes tiempos de procesamiento, como pueden ser: cálculos científicos, compilación de programas, compresión de datos, descifrado de códigos, renderización de imágenes.

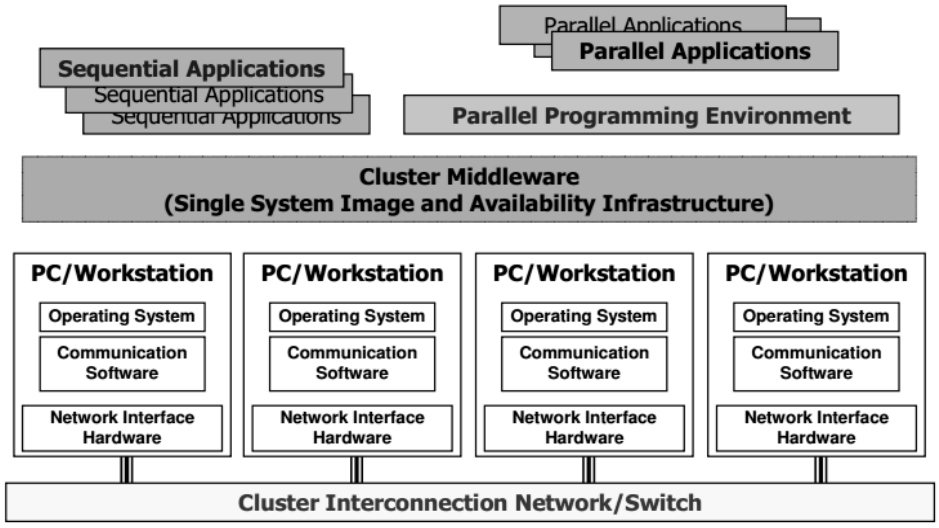

Figura 6: Arquitectura de un Cluster HP[8].

La arquitectura típica de un cluster de alto rendimiento se muestra en la Figura 6. Los componentes claves del cluster incluyen a los nodos los cuales cuentan con un sistema operativo, un software de comunicación y una interfaz de red, todos los nodos se encuentran interconectados por medio de un dispositivo de red (Switch) estos son vistos y administrados como una sola entidad gracias al Middleware.

Su funcionamiento se basa en la división de un problema en partes pequeñas, las cuales son procesadas por cada uno de los nodos que componen el cluster. Dentro de esta división existen 2 soluciones, las que funcionan a nivel aplicación y las que funcionan a nivel sistema, en las primeras para que una aplicación pueda aprovechar el poder de procesamiento de varios nodos

[8]R. Buyya (ed.), High Performance Cluster Computing: Architectures and Systems, vol. 1, Prentice Hall, 1999

debe de estar codificada para funcionar en paralelo usando las librerías correspondientes, esto genera inconvenientes, en caso de que la aplicación no estuviera diseñada para este fin y se tendrá que recodificar. En las segundas no es necesario recodificar las aplicaciones del usuario, sino que el propio núcleo del sistema operativo es el que se encarga de distribuir el trabajo de forma transparente a dicha aplicación, esto también conlleva problemas ya que el núcleo se vuelve mucho más complejo y es más propenso a fallos.

Las aplicaciones que son ejecutadas dentro del cluster son divididas en procesos pequeños y cada proceso es ejecutado en un nodo distinto, para conseguir esto existen dos formas[9]:

- Asignar estáticamente cada proceso a un nodo en particular: Se elige de forma manual el nodo en donde el proceso será ejecutado durante toda su vida, debido a esto es importante seleccionar de forma correcta estos nodos. Hay que tener en cuenta que en caso de un fallo en el algoritmo de elección de nodo este puede infrautilizar mucho de los recursos.
- Asignar dinámicamente procesos a los nodos: Los procesos pueden migrar de nodo a nodo dinámicamente. Dentro de este tipo de configuración es importante la política de localización y la política de migración. Aquí no es necesaria la configuración manual, cuando se desbalancea el sistema éste se encarga de que se vuelva a balancear, de tal forma de que se aprovechen los recursos al máximo.

Los procesos son ejecutados de manera colaborativa y sincronizada gracias al paso de mensajes entre ellos. Existen 2 clases de paso de mensajes: síncronos y asíncronos; en el paso de mensajes asíncrono, el proceso que envía, no espera a que el mensaje sea recibido, y continúa su ejecución, siendo posible que vuelva a generar un nuevo mensaje y a enviarlo antes de que se haya recibido el anterior.

[9] miKeL a.k.a.mc2, Kris Buytaert, http://ibiblio.org/pub/linux/docs/LuCaS/Manuales-LuCAS/doc-manual-openMosix-1.0/doc-manual-openMosix_html-1.0/node18_ct.html

En el paso de mensajes síncrono, el proceso que envía el mensaje espera a que un proceso lo reciba para continuar su ejecución[10].

Mediante este proceso los clusters de alto rendimiento reducen de forma considerable el tiempo en la obtención de resultados de cálculos complejos, los cuales serían imposibles resolver a través computo tradicional.

A continuación se detallara de forma extensa el funcionamiento de los clusters de alta disponibilidad y los clusters de balanceo de carga, esto debido a que son los tipos de clusters que se utilizaran en el resto de la tesis.

2.3 Clusters de alta disponibilidad (HA)

Antes de explicar el funcionamiento de este tipo de cluster, es necesario detallar el fin principal que se busca, al momento de implementar este tipo de tecnología.

La disponibilidad es la medida de tiempo en el que un servidor o proceso funciona de manera normal, también es la medida de tiempo necesario para recuperar el servicio después de un fallo en el mismo[11]. Alta disponibilidad, es un sistema diseñado e implementado para proporcionar acceso a los datos de forma continua 24 horas del día, 7 días a la semana y 365 días al año.

2.3.1 Matriz de disponibilidad

Para ciertas aplicaciones, el 99.9% de uptime es adecuado, esto deja un downtime de solo 1.4 minutos por día o 8.8 horas por año, dentro de los esquemas de alta disponibilidad es común

[10]http://es.wikipedia.org/wiki/Interfaz_de_Paso_de_Mensajes

[11] IBM http://www.redbooks.ibm.com/redbooks/pdfs/sg248168.pdf

referirse a estos porcentajes como dos nueves, tres nueves, cuatro nueves y cinco nueves, según el nivel de disponibilidad que otorgue el sistema.

Un Sistema de alta disponibilidad de cinco nueves dentro de las organización es generalmente el más aceptado ya que logra tiempos de downtime de tan solo 5 minutos al año y a un razonable costo.

Disponibilidad en porcentaje	Tiempo apoximado de perdida al año
99.9999% (seis nueves)	32 segundos
99.999% (cinco nueves)	5 Minutos
99.99% (cuatro nueves)	53 Minutos
99.90%	8.8 Horas
99.00%	87 Horas (3.6 dias)
90.00%	876 Horas (36 dias)

Tabla 1: Alta disponibilidad

2.3.2 Niveles de disponibilidad

Las soluciones de alta disponibilidad requieren una intensa planeación y continuo monitoreo, muchos componentes deben de ser consideradas al momento de diseñar e implementar una arquitectura HA. Por ejemplo:

- Servidores
- Bases de Datos
- Almacenamiento
- Redes de Comunicación
- Firewalls
- Sistema Operativo
- Servicios y procesos
- Aplicaciones Web

Existen múltiples tecnologías que pueden ser utilizadas para proporcionar alta disponibilidad a los componentes de una arquitectura. La disponibilidad puede ser divida en los siguientes 5 niveles[12]:

Nivel 1: Básico.- Dentro de este nivel no existe redundancia ni servicios de HA.

Nivel 2: RAID x.- Consiste en Arreglos básicos de discos, protege los datos en caso de daño físico en el disco, pero no da disponibilidad al servicio.

Nivel 3: Failover.- Nodos y servicios redundantes, si un nodo falla otro proporciona el servicio.

Nivel 4: Replicación.-Datos redundantes entre nodos, esto da una mejor protección en caso de falla en los discos o corrupción de datos.

Nivel 5: Recuperación de desastres.- En caso de un desastre natural, en donde se pierda toda la infraestructura de algún lugar, se continua con el servicio ya que se cuenta con infraestructura redundante. En la figura siguiente se muestran los niveles de HA.

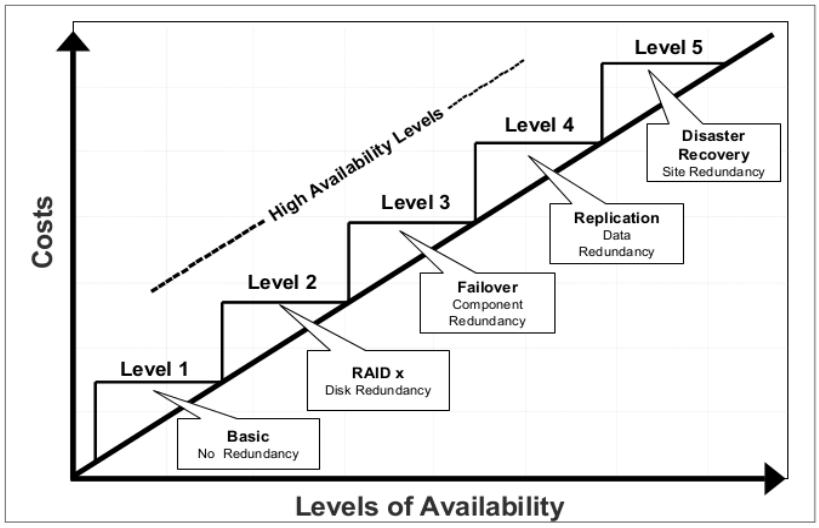

Figura 7: Niveles de disponibilidad.

[12] IBM http// www.redbooks.ibm.com/redbooks/pdfs/sg246688.pdf

Para integrar todos los elementos antes descritos en un solo sistema, se hace uso de un cluster de alta disponibilidad. Este tipo de cluster ofrece el mayor tiempo posible de disponibilidad a algún servicio, para esto generalmente se auxilia de equipos redundantes los cuales están a la espera en caso de que ocurra alguna falla con el servicio original.

El concepto básico de un cluster de alta disponibilidad es conocido como faillover. Cuando algún nodo del cluster falla, otro miembro de este, deberá de comenzar a hacer el trabajo del nodo que presento el problema, este proceso, no es sencillo, es necesario tomar en cuenta algunos aspectos:

1. ¿Cómo sabe un nodo que a otro le ocurrió alguna falla?
2. ¿Cómo se puede acceder a los últimos datos que el nodo fallido estaba utilizando?
3. ¿Cómo sabe el usuario a que nodo conectarse después de que ocurrió alguna falla?

La implementación de un cluster en alta disponibilidad no es difícil, pero si compleja ya que se debe considerar de igual forma una lista grande de variables.

2.3.3 Heartbeat

Dentro de esta clase de clusters existe un continuo envío de mensajes de estado entre los nodos, a este paso de mensajes se le conoce como heartbeat, los mensajes de heartbeat indican al receptor, que el nodo que envió el mensaje se encuentra funcionando correctamente. Cada miembro del cluster espera recibir este mensaje en un periodo de tiempo límite, el nodo que no envíe su heartbeat dentro de este rango de tiempo se le considerara muerto.

El continuo heartbeat entre nodos presenta algunas dificultades al momento de lograr el alta disponibilidad en el cluster, cada nodo por diversas razones, como una latencia anormal en la red, o ejecución anormal del proceso de heartbeat, puede ocasionar que el mensaje no llegue en el tiempo adecuado y se considere al nodo como fuera de línea sin en realidad estarlo, este problema se presenta con frecuencia en ambientes reales, se detallará más a fondo el problema con los heartbeats en la sección *2.5 Problemas en Clusters.*

Protocolo HeartBeat

Un protocolo heartbeat permite a procesos dentro del mismo programa en una red, intercambiar periódicamente mensajes de pulsaciones.

Mientras que el proceso ***p*** permanezca recibiendo mensajes de pulsaciones de parte del proceso ***q***, el proceso ***p*** reconoce que el proceso ***q*** y el medio de comunicación de ***q*** a ***p*** están ambos disponibles. Si no recibe ninguna pulsación de parte de ***q*** por ***a*** periodo de tiempo, ***p*** reconoce que ***q*** a fallado o que la comunicación entre ***q*** y ***p*** fallo, en este caso ***p*** se termina a si mismo [13]

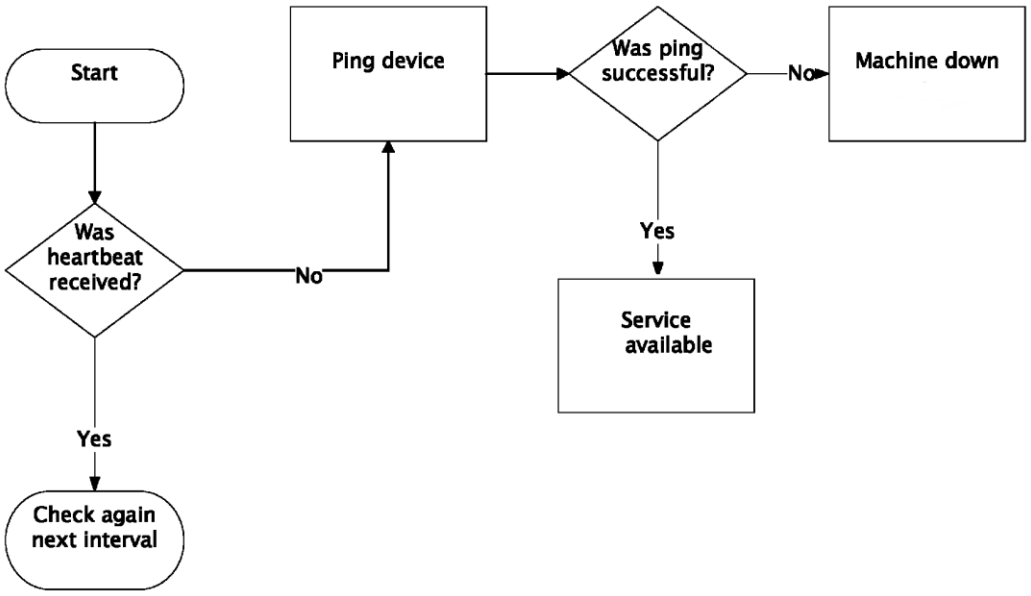

Figura 8: Diagrama de flujo del funcionamiento de un heartbeat en cluster.

[13]Mohamed G. Gouda ,Tommy M. McGuire, Accelerated Heartbeat Protocols https://webspace.utexas.edu/goudamg/IEEE/00679503.pdf

2.3.4 Recursos en ambiente de HA

Se le llama recurso a cualquier elemento que le sea posible pasar de un nodo a otro en caso de falla dentro del cluster, por ejemplo: instancias de aplicaciones, direcciones IP y dispositivos de almacenamiento, estos recursos pueden depender unos de otros, por lo tanto el cluster debe de conocer las dependencias de estos. Si no se establece una correcta política de dependencia, pueden surgir problemas, un usuario por ejemplo puede perder información, escribir nuevos datos en el espacio reservado para otro usuario o sencillamente no poder acceder a su aplicación.

Para mantener un correcto seguimiento de los recursos y sus dependencias, generalmente dentro del cluster debe de existir alguna herramienta que permita agregar, editar y eliminar la información del recurso.

Otro aspecto importante es la forma en que los recursos inician en los nodos, esto se logra con un agente de recursos, la función de este software es la de iniciar, detener y monitorizar la aplicación a la que se le desea dar alta disponibilidad, a casi cualquier aplicación se le puede otorgar esta característica sólo es necesario un agente de recursos que la administre.

Una vez que un nodo se ha declarado fuera de línea se debe de comenzar con la reubicación de los recursos, en un cluster de 2 nodos la reubicación de recursos es sencilla ya que solo pasa de un nodo a otro, pero dentro de un cluster de más de 3 elementos la reubicación se vuelve complicada, se pueden considerar las siguientes soluciones:

- Que el usuario defina que nodo es mejor según sus características físicas para alojar el recurso.
- Una lista predefinida por el administrador del sistema con los nodos disponibles y el lugar que ocuparan en la reubicación.
- Distribuir los recursos en todos los nodos restantes de manera que cada nodo tenga la misma carga de trabajo.

2.3.5 Alta disponibilidad en el almacenamiento de datos

El aspecto más crítico de un esquema de alta disponibilidad es el nivel de protección que se debe de aplicar al almacenamiento, ya que en caso de que una aplicación falle, es posible sustituirla de forma transparente y rápida, no así con la pérdida o corrupción de datos, la cual de presentarse representa un serio problema.

La forma tradicional de almacenamiento de datos en servidores productivos es en forma de discos duros, estos discos pueden presentar fallas físicas o lógicas. Existen soluciones que intentan minimizar estos problemas:

RAID Local

RAID en inglés Redundant Array of Inexpensive Disks provee mecanismos para que los datos se encuentren disponibles y legibles aun en caso de alguna falla en disco. El sistema administra los discos que se encuentran dentro de un RAID asignando la característica a alguno de estos discos de contener la información necesaria para conocer el contenido de algún disco dañado.

Normalmente una suma de bits de todos los discos está guardada en algún disco destinado a redundancia, para que en caso de que se presentara alguna falla en disco, se pueda realizar el cálculo de la información perdida. (RAID Nivel 5). En la siguiente tabla se muestran los niveles de RAID.

Category	Level	Description	Disks required	Data availability	Large I/O data transfer capacity	Small I/O request rate
Striping	0	Nonredundant	N	Lower than single disk	Very high	Very high for both read and write
Mirroring	1	Mirrored	$2N$	Higher than RAID 2, 3, 4, or 5; lower than RAID 6	Higher than single disk for read; similar to single disk for write	Up to twice that of a single disk for read; similar to single disk for write
Parallel access	2	Redundant via Hamming code	$N + m$	Much higher than single disk; comparable to RAID 3, 4, or 5	Highest of all listed alternatives	Approximately twice that of a single disk
	3	Bit-interleaved parity	$N + 1$	Much higher than single disk; comparable to RAID 2, 4, or 5	Highest of all listed alternatives	Approximately twice that of a single disk
Independent access	4	Block-interleaved parity	$N + 1$	Much higher than single disk; comparable to RAID 2, 3, or 5	Similar to RAID 0 for read; significantly lower than single disk for write	Similar to RAID 0 for read; significantly lower than single disk for write
	5	Block-interleaved distributed parity	$N + 1$	Much higher than single disk; comparable to RAID 2, 3, or 4	Similar to RAID 0 for read; lower than single disk for write	Similar to RAID 0 for read; generally lower than single disk for write
	6	Block-interleaved dual distributed parity	$N + 2$	Highest of all listed alternatives	Similar to RAID 0 for read; lower than RAID 5 for write	Similar to RAID 0 for read; significantly lower than RAID 5 for write

N = number of data disks; m proportional to log N

Tabla 2: Niveles RAID.

Dispositivos de Bloque en Red

Éste dispositivo es un equipo más conectado a la red, es visto por los servidores como un disco local, el cual puede ser particionado, formateado y usado. Existen distintas soluciones en el mercado que proveen este tipo de dispositivos, generalmente proveen una interfaz web en donde es posible administrar los discos y la forma en la que la información es almacenada en ellos. El protocolo más usado para la presentación de los dispositivos de bloque es el ISCSI

Sincronización de datos

Para que una aplicación ofrezca alta disponibilidad no basta con que ésta se inicie en distintos nodos, el punto más importante es que los datos de la aplicación de igual manera sean reubicados y se encuentren actualizados con los últimos datos escritos antes de que el nodo fallara.

Existen 2 formas para lograr esto:

- Usando Almacenamiento compartido dentro del cluster, este tipo de configuración requiere que cada miembro del cluster esté conectado a un disco el cual contenga los datos de la aplicación.
- Utilizando un espejo remoto, esta configuración usa la red del usuario para replicar los datos de un nodo a todos los demás.

Ambas tecnologías tienen ventajas y desventajas según sus características, las cuales se exponen en la siguiente tabla comparativa:

Características	**Almacenamiento Compartido**	**Espejo Remoto**
Costo	Generalmente se utilizan conexiones propietarias o de fibra óptica lo cual representa un precio mayor, además de requerir adaptadores SCSI y tarjetas controladoras.	Requiere de una conexión Ethernet, algún tipo de disco para el almacenamiento en cada nodo y un software que se dedique a realizar las replicas.
Distancia entre Nodos	Según el tipo de conexión se puede lograr una distancia ilimitada teórica utilizando repetidores.	No existe límite de distancia entre nodos.
Sincronización	La sincronización se realiza de forma inmediata	Existen varios factores que pueden provocar que la sincronización de datos consuma tiempo.
Tolerancia a Fallos	Al estar alojado en el mismo site, si existe algún incidente, no cuenta con ninguna protección.	Como la información se encuentra en cada uno de los nodos, es posible lograr tolerancia a fallos incluso si en el site ocurriera algún desastre.
Rendimiento	El rendimiento de lectura/escritura en un sistema de Almacenamiento compartido es igual a la velocidad en un sistema local.	El espejo remoto usa un disco local para leer las operaciones, por lo tanto el rendimiento de lectura es el mismo, la velocidad de escritura puede verse afectada ya que todos los nodos deben de replicar los datos.

Tabla 3: Comparación entre almacenamiento compartido y espejo remoto.

Recuperación de datos

La redundancia de datos es fundamental para evitar corrupción o perdida de información, incluso con un esquema de alta disponibilidad en donde existen soluciones como RAID o Dispositivos de Almacenamiento, es aconsejable que se realicen copias de los equipos más críticos, por ejemplo Bases de Datos. Existen 2 estrategias principales para el respaldo de la información:

1. Respaldos versionados

   Estos respaldos generalmente se realizan diariamente en horarios no productivos, donde la carga de trabajo es la mínima y es posible realizar copias hacia un sitio remoto sin riesgo de afectar al rendimiento de la producción. Los respaldos son enviados a equipos en algún punto geográfico distinto, para así asegurar que en caso de algún desastre natural en el sitio principal, exista un respaldo en un sitio secundario.
2. Recuperación rápida

   Es usada para contrarrestar pequeñas perdidas de datos, puede existir por ejemplo un equipo activo y otro pasivo, el ultimo se encuentra replicando los datos del activo, logrando con esto que si el sistema activo falla automáticamente se redirija al equipo pasivo sin perder la información entrante.

Sincronización asíncrona y sincronización síncrona

La diferencia entre replicación síncrona y asíncrona, es que la primera garantiza que en caso que existiera alguna modificación en algún nodo del cluster esta ocurrirá en los otros nodos al mismo tiempo. En cambio la replicación asíncrona no garantiza que las modificaciones ocurran en todos los nodos al mismo tiempo, generalmente esto ocurre con un pequeño retraso de tiempo.

2.3.6 Alta disponibilidad en la red de datos

Para lograr un correcto esquema de alta disponibilidad es necesario también tomar en cuenta los equipos de comunicación que utiliza el cluster para conectarse, ya que si se presenta alguna falla en ellos, aun cuando se tenga configurada una correcta redundancia en servidores y almacenamiento, el sistema quedara inutilizable.

Dentro de una comunicación entre nodos existen varios puntos de fallo, estos puntos se clasifican en 3 categorías o niveles:

Nivel de Link: Es usado para proveer redundancia a los adaptadores de red y switches. Considerando que la tarjeta de red puede fallar, un servidor dentro de este esquema debe de tener más de una tarjeta para asegurar la conectividad, también es posible que se presenten fallas en el switch de comunicación por lo que es recomendable que se tengan dos o más switches en la misma red, y los servidores conectados a más de uno de ellos, para que en caso de falla, el switch restante siga proporcionando conectividad.

Nivel de Gateway: Todos los servidores se conectan a un Gateway para así poder comunicarse con otros equipos fuera de su red, dentro de un esquema de alta disponibilidad cada Gateway debe de contar con múltiples tarjetas de red, además de routers redundantes los cuales comparten una dirección IP virtual , según el fabricante , se ofrecen distintos protocolos de heartbeat para comprobar el estado de todos los routers y en caso de daño físico en uno de ellos, los routers restantes continúan con el servicio.

Nivel de Routers: Este tipo de esquema se usa para la redundancia de redes extensas, en donde debido a las rutas tan complejas de los paquetes no es práctico solo contar con redundancia en Gateways. Para lograr alta disponibilidad a este nivel, todas la rutas son propagadas a todos los routers, para que en caso de que algún router falle, los demás conozcan la ruta que el paquete debe de tomar.

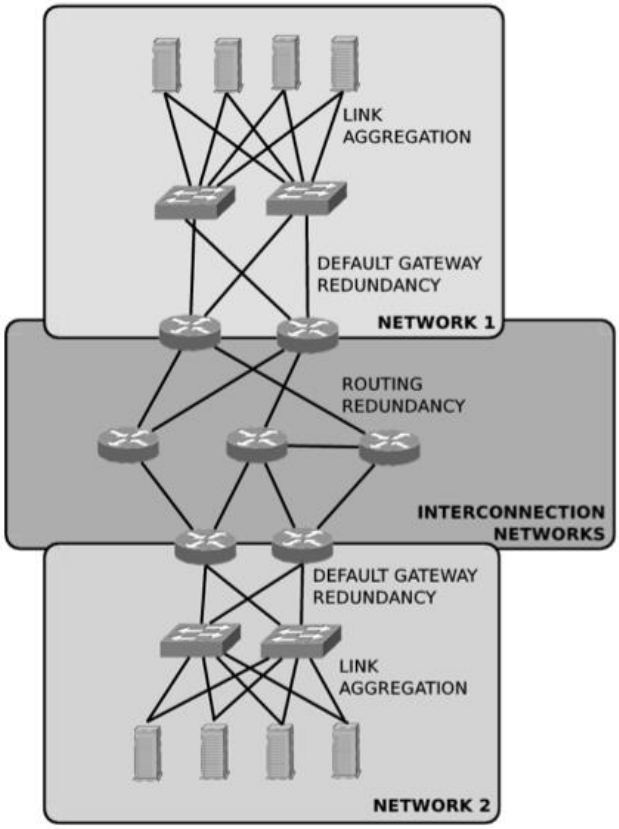

Figura 9: Diagrama de una red en alta disponibilidad.

2.3.7 Alta Disponibilidad en Software

Es fundamental considerar que todo software contiene fallas de programación, estas pueden ocasionar comportamientos inesperados durante la ejecución de procesos y provocar fallas de estabilidad, lo que afecta la disponibilidad del recurso que se ofrece. Es posible dividir las fallas que sufre el software en las siguientes categorías:

- Configuración – Fallas en la configuración del servidor, parámetros mal establecidos, un uso incorrecto de las, mejores prácticas de implementación del software.
- Diseño – Este tipo de fallas son introducidas por el desarrollador de la aplicación y son errores en el código o comportamiento erróneo en ambientes de trabajos inesperados.
- Ambiente – Son ocasionados por factores independientes a la aplicación, por ejemplo librerías del sistema operativo, falta de dependencias en paquetes.
- Ciclos Infinitos – En estos errores, el fin de un ciclo no es bien definido por parte del desarrollador y provoca el uso de recursos en exceso por parte de proceso.
- Algoritmo Insuficiente – También es introducido por el desarrollador y si bien no provoca un error, usa de forma inadecuada los recursos, causando lentitud en la aplicación.
- Concurrencia – El software no es creado o configurado para soportar grandes números de peticiones lo que provoca, la falla o crash de procesos.
- Error de Usuario – Estos errores son causados por el usuario final, y provocados por no tener el conocimiento necesario del aplicativo que se usa.

En un ambiente de alta disponibilidad es necesario eliminar en lo posible las fallas a nivel software, a través de distintos métodos, principalmente los procesos críticos son monitoreados y en el momento de detectarse algún problema en un proceso se alerta al administrador para que se valide manualmente y se aplica un reset al proceso, esto puede ser de forma automática según las políticas establecidas.

No solo el software de aplicación puede fallar, el software base también es un elemento crítico, Linux es considerado actualmente como el mejor sistema operativo para servidores de aplicaciones, dada su facilidad de adaptación a todos los ambientes, su rapidez y estabilidad.

Sin embargo en ambientes críticos es necesario dar un tratamiento especial al sistema operativo, ya que fallas de comunicación con el hardware, reinicios inesperados provocados por procesos bloqueados, actualizaciones fallidas y Kernel Panics son algunos de los

problemas comunes que Linux presenta en ambientes de alta demanda y una arquitectura en alta disponibilidad debe de estar preparada para soportar también este tipo fallas.

2.3.8 Arquitectura interna de un software HA

Anteriormente se explicó cómo se puede proporcionar alta disponibilidad a los elementos que componen una arquitectura típica dentro de una organización, como ya mencionamos para administrar todos estos recursos de forma correcta se requiere de un software especial el cual le proporciona la característica de alta disponibilidad al cluster de computadoras.

Este software se ejecuta a manera de servicio en cada uno de los nodos del cluster, se encuentra monitoreando y recibiendo mensajes entre procesos, su arquitectura interna es la siguiente[14]:

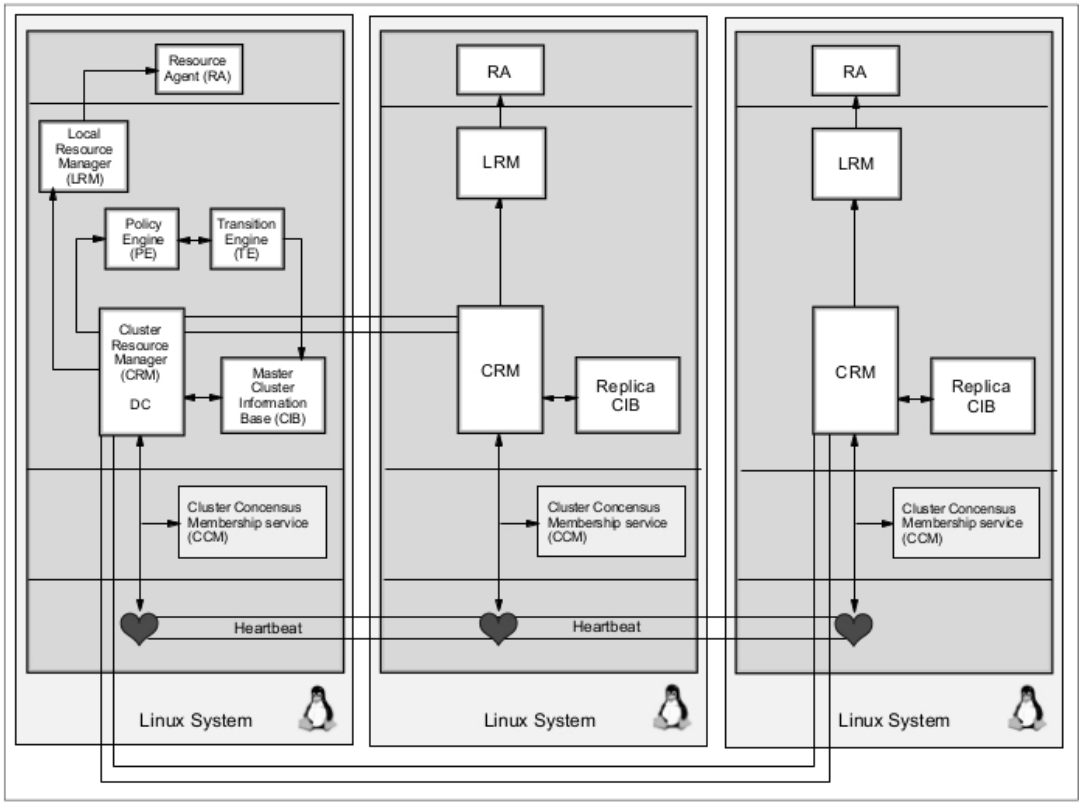

Figura 10: Capas de un servicio tipo Heartbeat ejecutándose en 3 nodos.

[14] SUSE HA https://www.suse.com/documentation/sle_ha/pdfdoc/book_sleha.pdf

Capa de Mensajes e Infraestructura

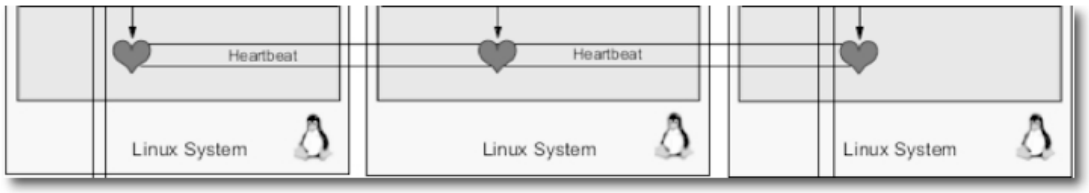

Figura 11: Capa de mensajes Heartbeat

Dentro de esta capa se encuentran los componentes que hacen uso del protocolo Heartbeat, envían mensajes del tipo "estoy vivo" a los otros miembros del cluster para que así conozcan el estado de cada uno de ellos, todas las comunicaciones entre los nodos del cluster se realizan en esta capa. El componente de heartbeat es crucial para el funcionamiento correcto del cluster, por lo que el canal de comunicación por el que se envían las señales de heartbeat debe de estar siempre disponible.

Capa de Afiliación

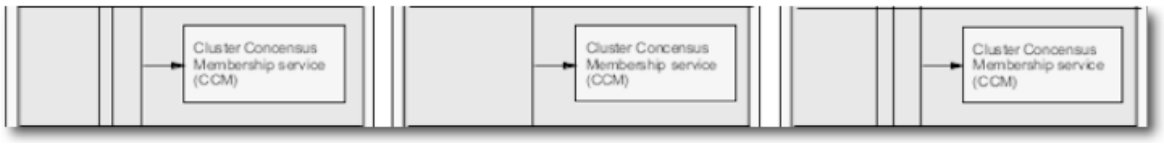

Figura 12: Capa de afiliación

La capa de afiliación es responsable de calcular y conocer a todos los nodos que se encuentran conectados y son parte del cluster, esto lo realiza en base a la información que se obtiene de la capa de mensajes, el servicio interno que se encarga de esta tarea es el CCM (Cluster Consensus Membership), el cual provee y organiza la topología actual del cluster, la cual será usada por los componentes en las capas superiores.

Capa de Asignación de Recursos

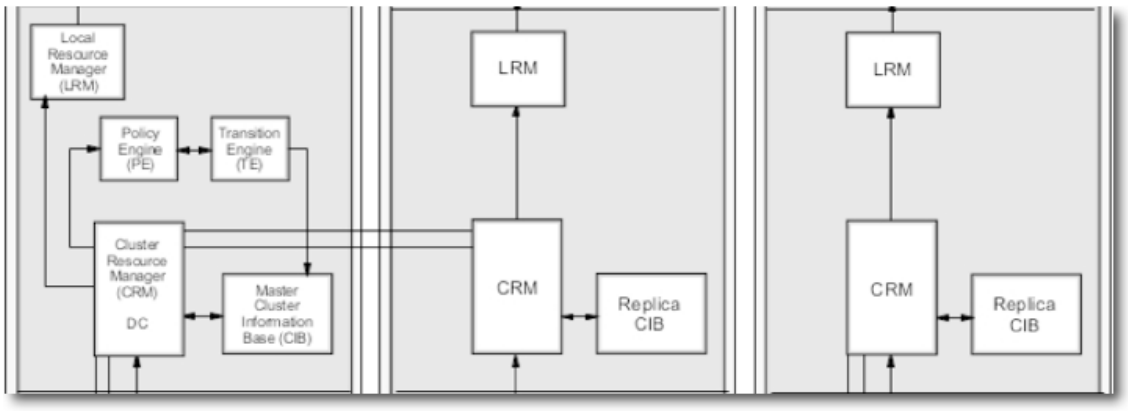

Figura 13: Capa de asignación de recursos

En esta capa se establecen y almacenan todas las reglas del cluster, los componentes dentro de esta capa toman decisiones, en base a reglas establecidas al momento de ocurrir ciertos eventos. Existen varios componentes dentro de esta capa, a continuación son descritos:

- **Base de Información del cluster (CIB)** El CIB es una representación entera de la configuración del cluster y su estatus, incluye a todos los nodos miembros así como sus recursos.

  Dentro del cluster existe un nodo maestro CIB, y todos los nodos restantes mantienen un replica de este, el administrador puede manipular el comportamiento del cluster mediante órdenes en línea de comando o interfaz gráfica en el nodo maestro.
- **Administrador de Recursos del Cluster (CRM)** Cada nodo en el cluster cuenta con su propio CRM, que contiene su réplica CIB y se comunica con el administrador local de recursos (LRM) el cual ejecuta al Agente de Recursos (RA) además también se comunica con la capa de Mensaje. El CRM es el encargado de procesar todas las transacciones que pasan por la capa de asignación de recursos. Un nodo dentro del cluster (Nodo maestro) es elegido como Coordinador designado (DC). El DC debe de mantener al CIB maestro y comunicar los cambios a los otros CRMs que corren en los demás nodos.

- **Motor de Políticas y Motor de Transición** El Motor de Políticas (PE) se encarga de establecer los procesos que se deben de iniciar en el momento que uno de los nodos falla. Los comandos que el PE indica son luego ejecutados por el motor de transición (TE).
- **Administrador Loca de Recursos** El administrador Local de Recursos de encarga de mandar llamar a los distintos agentes. Pueden iniciar, detener o monitorear las operaciones de los agentes y después informarlo al CRM.

Capa de Recursos

Figura 14: Capa de recursos

Esta capa es la última dentro de la arquitectura de un servicio de alta disponibilidad y contiene a los agentes de recursos que controlan las operaciones hacia los servicios. A cada recurso que se le quiera agregar alta disponibilidad se le asocia un agente que controla sus operaciones.

2.3.9 Flujo de un procesos en HA

Distintas acciones pueden provocar cambios en el cluster como por ejemplo:

- Un recurso o nodo que falle.
- Agregar o remover algún recurso del nodo.
- Iniciar o detener un recurso.
- Añadir o Remover un nodo.
- Migrar un recurso de un nodo a otro.

A continuación se describe el flujo de los procesos durante un cambio de recursos en el cluster, producido por la falla en uno de sus nodos:

1. Si no se producen beats de alguno de los nodos en un tiempo determinado, se le es informado al CCM.

Figura 15: CCM reconoce una falla en cluster

2. El CCM envía paquetes a sus pares en el cluster y determina exactamente cuales nodos continúan dentro del cluster y cuáles no.

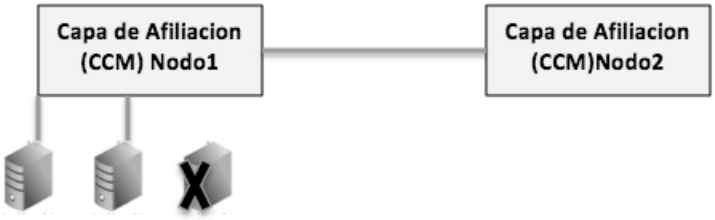

Figura 16: CCM actualiza el estado del cluster

3. El CCM notifica al CRM en el DC dentro de la capa de alocación de recursos.

Figura 17: CRM conoce el estado del cluster

4. El CRM actualiza su CIB maestro e informa sus CRMs pares.

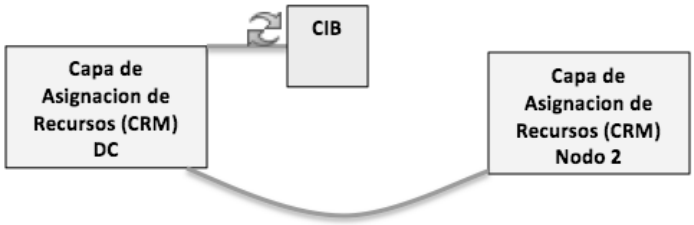

Figura 18: CRM actualiza el nuevo estado del cluster

5. El CRM notifica al PE, este último busca el CIB y observa que cambios deben de realizarse dentro del cluster.

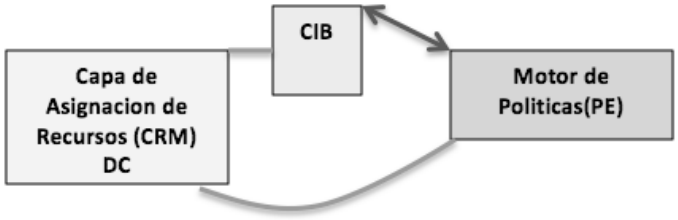

Figura 19: PE obtiene lista de cambios a realizar

6. El PE genera una lista de acciones a seguir según las políticas del cluster y se la envía al CRM. El CRM pasa la lista al TE para ejecutarlas.

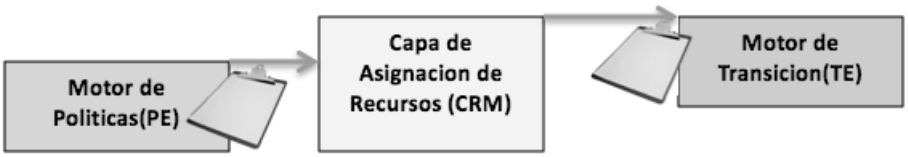

Figura 20: TE ejecuta lista de cambios

7. El DC envía mensajes a los CRMs relevantes en el cluster, los cuales usan sus LRM para realizar los cambios locales necesarios.

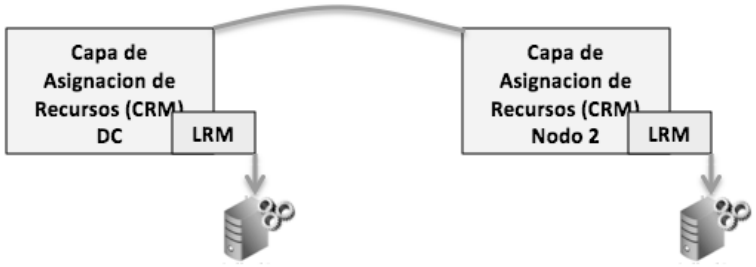

Figura 21: Ejecución de cambios locales

8. El DC notifica que las acciones han concluido.

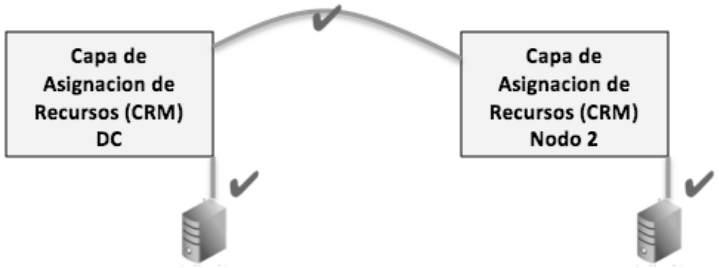

Figura 22: DC informa del nuevo estado del cluster

9. El cluster regresa a un estado de espera hasta que el un evento nuevo ocurra.

2.3.10 Tipos de configuraciones en alta disponibilidad

Activo/Activo

Usando esta configuración, todos los servidores en el cluster pueden ejecutar simultáneamente los mismos recursos. Los clientes pueden acceder independientemente a cualquiera de los nodos, en caso de alguno de los nodos no esté disponible, los recursos se encuentran aún disponibles en el otro nodo. Una de las ventajas de usar esta configuración es que el cluster es

más eficiente debido a que todos los nodos están recibiendo todas las peticiones al mismo tiempo.

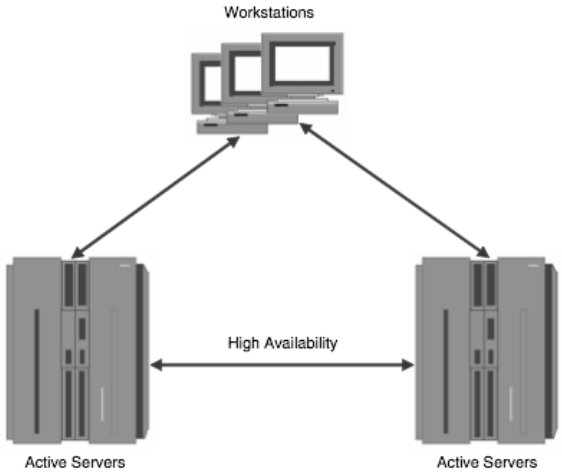

Figura 23: Arquitectura Activo/Activo.

Activo/Pasivo

Esta configuración consisten en un servidor que se encuentra ejecutando el recurso destinado a alta disponibilidad, y los demás se encuentran a la espera, en caso de que este nodo fallara, los nodos restantes iniciarían el recurso y responderían a las peticiones de los clientes.

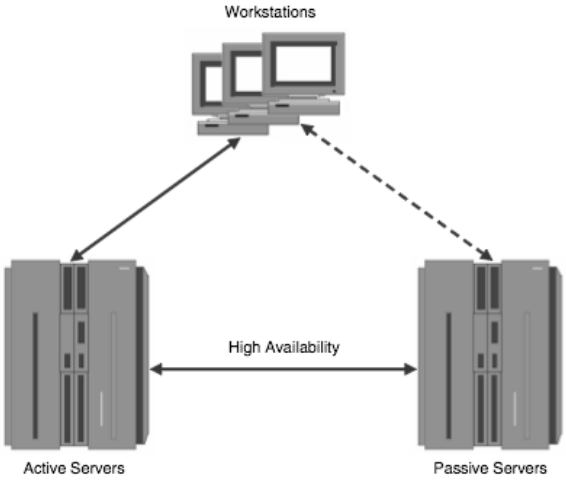

Figura 24: Arquitectura Activo/Pasivo.

2.3.11 Funcionamiento de un cluster HA

Después de haber explicado el comportamiento interno de un software de HA así como de cada uno de los procesos que se realizan para administrar la falla de en uno de sus componentes, mostraremos ahora el funcionamiento completo de un cluster a nivel arquitectura durante un evento de failover.

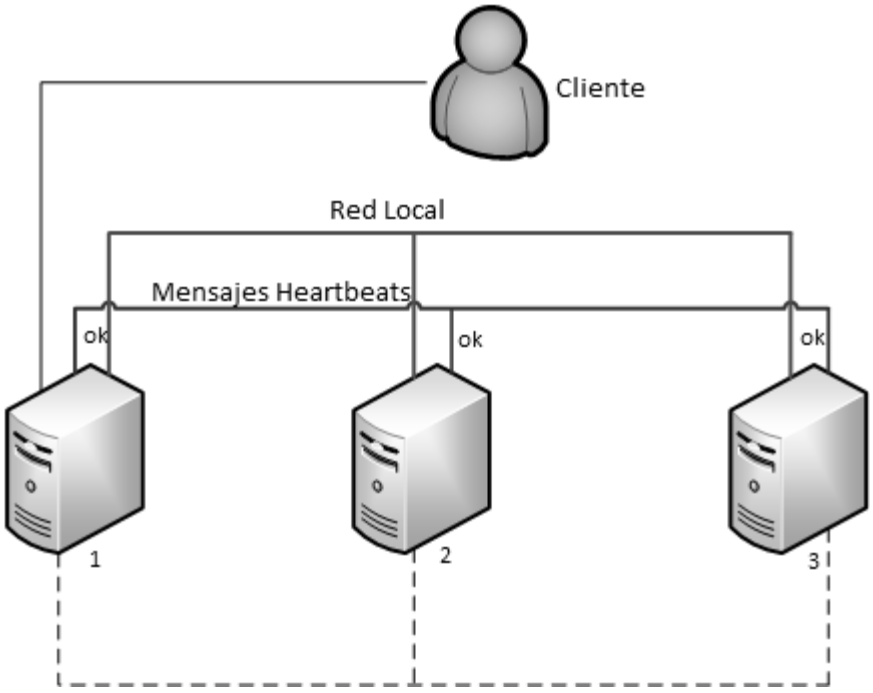

Figura 25: Funcionamiento de un cluster HA.

Dentro de este ambiente de ejemplo, el **servidor 1** está ofreciendo servicios a los clientes que se conecten a él, al mismo tiempo está mandando mensajes heartbeat a los **servidores 2 y 3** los cuales también envían su propio heartbeat, los datos que generen las aplicaciones que el cliente está utilizando se replicaran de forma automática a los **servidores 2 y 3.** *Figura 25*

Ahora se asume que el **servidor 1** sufre un fallo y ya no envía mensajes heartbeat a los otros dos nodos dentro del cluster. *Figura26*

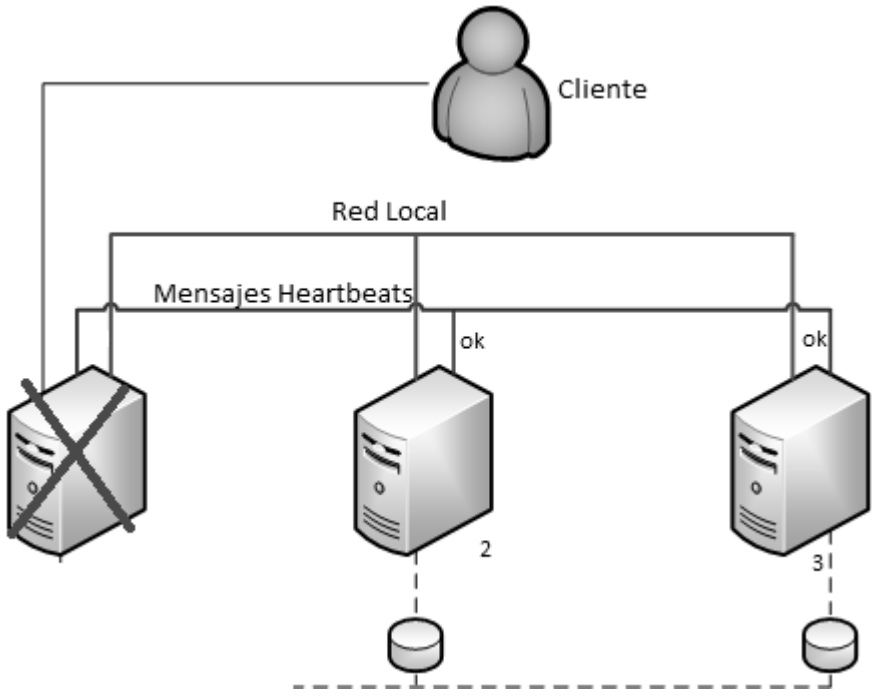

Figura 26: Funcionamiento de un cluster HA.

Debido a la falta de heartbeats por parte del **servidor 1** durante algún tiempo establecido, el software de HA del **servidor 2** comienza a reubicar los servicios, datos e IP del servidor.

Cuando el cliente intente acceder a su aplicación, ahora el **servidor 2** contestara su petición, sin que el cliente se entere del proceso que ocurrió dentro del cluster ya que comparte la misma dirección IP virtual. *Figura27*

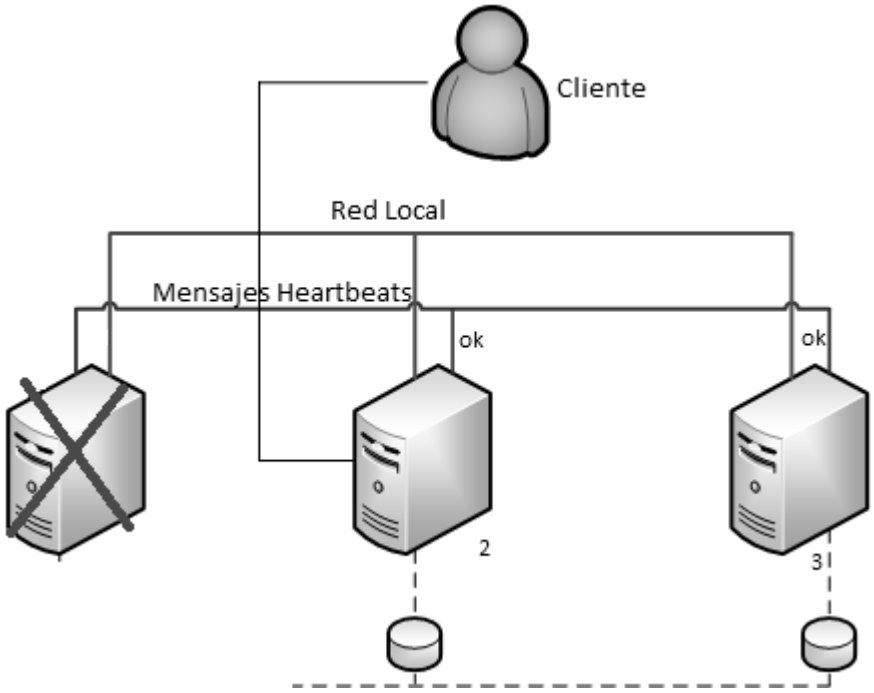

Figura 27: Funcionamiento de un cluster HA.

Mientras tanto el **servidor 3** sigue escuchando los heartbeats del **servidor 2** y en caso de falla realizara las misma acciones, cabe mencionar que el **servidor 2** contiene los datos actualizados que el **servidor 1** generaba por medio de la replicación datos.

Failback

Anteriormente se habló sobre el proceso de reubicación de recursos dentro del cluster al momento de fallar uno de sus nodos, su proceso inverso restaura los recursos al nodo en el que originalmente se encontraban, a esto se llama failback.

Durante el proceso de failback, el nodo que ha sufrido la falla comienza de nuevo a enviar mensajes heartbeat al agente, este se encarga de restaurar todos los servicios que estaban corriendo antes de la falla además de la sincronización de los datos del disco duro, una vez que se ha terminado con este proceso el nodo comienza a aceptar y responder peticiones de clientes, esto considerando que el nodo que sufrió la falla sea en nodo maestro.

Es importante que el nodo que se vuelve a incorporar al cluster después de que sufrió una falla tenga actualizados los datos que se pudieron generar en el tiempo que se mantuvo sin operar. Si esto no ocurre, se pueden presentar fallas en la aplicación e incluso inconsistencias en los datos.

Fencing o STONITH

Dentro de la tecnología de clusters el fencing o anteriormente llamado STONITH ("Shoot The Other Node In The Head") consiste en la aplicación de un reinicio o apagado forzado al nodo que presenta fallas, provocando con esto que sea ignorado por el agente de realojamiento de recurso y evitando problemas graves que pueden presentarse en el cluster y de los que se hablará más adelante, con esto se logra salvaguardar la integridad del cluster.

Este proceso se realiza con la ayuda de un dispositivo de fencing que en realidad se trata de una consola de administración remota dentro del servidor (ILO, IMM, IDRAC), y que el demonio de fencing puede comunicarse, ejecutando comandos como reset o shutdown según sea necesario.

2.4 Clusters de balanceo de carga (LB)

Este tipo de cluster cuenta con 4 usos principales[15]:

- Servidor de balanceo de carga.- Se usan para manejar la carga y distribuirla a través de múltiples nodos esto con el fin de escalar un servidor y para lograr tolerancia a fallos.
- Servidor global de balanceo de carga.- Se encarga de redireccionar a los clientes hacia data centers ubicados en zonas geográficas diferentes logrando con esto una mayor velocidad de respuesta y tolerancia a fallos en caso de algún desastre natural.

[15] Load Balancing Servers, Firewalls, and Caches. Chandra Kopparapu

- Balanceo de carga en firewalls.- Distribuye el tráfico hacia múltiples firewalls para escalar la capacidad de uno solo, y tolerar fallas físicas de algún firewall.
- Switcheo transparente de cache.- Redirecciona de forma transparente las peticiones hacia caches con contienen datos estáticos de la aplicación, esto con el fin de acelerar el tiempo de respuesta y mejorar el rendimiento del servidores Web.

Nos enfocaremos en el primer uso, ya que será este el que se implementara más adelante. Como se mencionó, éste tipo de cluster distribuye la carga provocada por las consultas de los clientes entre varios servidores para así lograr un equilibrio en el performance y tiempos de respuesta de la aplicación Web, son escalables ya que es posible agregar más servidores al cluster de forma transparente para el usuario.

Su arquitectura básica como se muestra en la *Figura 28* consiste en un equipo balanceador, y detrás de él, los nodos (Servidores Reales) donde está alojado el recurso, es el balanceador quien decide hacia que servidor mandar la petición, esto lo hace mediante un algoritmo de balanceo.

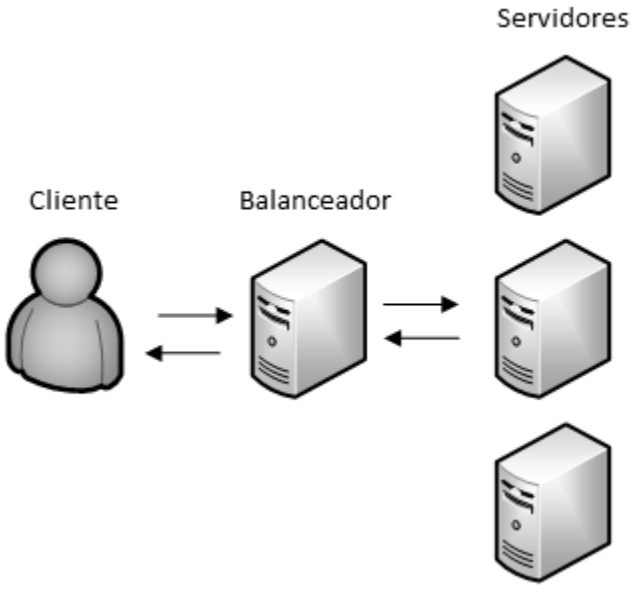

Figura 28: Arquitectura básica de un cluster de balanceo de carga.

Todos los servidores reales están conectados con el balanceador generalmente por un switch, el balanceador de carga es visto como un servidor virtual por los clientes, éste necesita una IP

para que los clientes se conecten a él, ésta IP es llamada IP Virtual. La IP virtual representa al cluster en su totalidad.

Un Cluster de Balanceo de carga una vez configurado ofrece varios beneficios a la infraestructura:

Escalabilidad: Como se mencionó anteriormente, es el balanceador el encargado de distribuir las peticiones a los servidores reales, por lo tanto la capacidad colectiva del servidor virtual es mayor que la del servidor real, una vez conformado el cluster es posible aumentar o disminuir el número de servidores reales dependiendo de la carga del entorno.

Disponibilidad: El balanceador de carga monitorea de forma continua el estado de los servidores reales, si detecta alguna falla en el canal de conexión que existe entre ellos, el balanceador deja de enviar peticiones hacia ese nodo, esto lo realiza de forma transparente para el usuario logrando así un aumento considerable al "uptime" de la aplicación.

Seguridad: Debido a que el balanceador esta frente a los servidores reales, actúa como una capa extra de seguridad, es posible implementar algún tipo de mecanismo de seguridad (IDS) en el equipo de balanceo, además de que generalmente la IP homologada se le es asignada al balanceador y los demás servidores reales continúan con su IP privada sin ser visibles desde Internet.

Calidad de Servicio: Esta se ve reflejada en que las respuestas a las peticiones de los usuarios son realizadas de manera más rápida, además que desde la vista de los clientes el servicio es prácticamente continuo sin caídas.

Un balanceador realiza la identificación del servidor al cual serán enviadas las peticiones en 2 partes:

1. El Balanceador de carga debe de identificar cuáles de los servidores reales están funcionando de forma correcta, mediante la realización de un chequeo de funcionamiento.
2. El Balanceador usa un algoritmo de distribución de carga para seleccionar al servidor, basado en las condiciones de carga actual que existe en los equipos

2.4.1 Checks de Funcionamiento

Los balanceadores de carga pueden realizar varios chequeos de funcionamiento en varias capas del Modelo OSI. En la capa 2, el balanceador envía una petición ARP a cada nodo, estos deben de contestar a menos que se encuentren abajo. En la capa 3, el balanceador realiza pings a los nodos en espera de que todos contesten de forma correcta. En la capa 4, el balanceador intentara conectarse a los servidores reales por el puerto específico de la aplicación que está corriendo.

También se pueden configurar para realizar tests específicos en la capa 7, por ejemplo si se tiene una aplicación corriendo por el puerto 8080, el balanceador puede lanzar periódicamente peticiones mediante una liga de prueba, en espera de algún resultado determinado, de no ser este el caso se considera que la aplicación no se encuentra activa.

2.4.2 Algoritmos de balanceo

Este tipo algoritmos se dividen en 2 clases, los estáticos que distribuyen uno a uno la carga, sin considerar el estado actual del servidor. Y los dinámicos que toman decisiones inteligentes en el envío de paquetes. Los más usados en el balanceo de carga son:

Round-Robin

Este algoritmo asigna un periodo de tiempo a cada flujo de datos y el envío de paquetes se hace de manera alternada uno a uno. Por ejemplo si se tuvieran 3 colas ***C1, C2,*** y ***C3*** a cada una se le asignaría un periodo de tiempo ***t1***. El algoritmo comenzaría a enviar los paquetes a

C1 durante un tiempo de ***t1***, después enviaría paquetes a ***C2*** en un tiempo ***t1***, y posteriormente los enviaría a ***C3*** en ***t1***, para después regresar a la cola ***C1*** y enviar paquetes nuevamente.

Weighted Round-Robin

Éste algoritmo es una versión modificada del round robin, en el cual a cada servidor se le asigna un peso, un numero entero el cual indica su capacidad de procesamiento, el algoritmo primero envía paquetes en mayor cantidad a los servidores que tienen un valor de peso más alto mientras que a los servidores con un valor igual o más bajo, envía un número igual de nuevos paquetes.

Por ejemplo:

Dado los servidores *A, B, C* con valores de peso *4, 3, 2* respectivamente, el algoritmo enviara los paquetes a los servidores de la siguiente forma *AABABCABC* durante un periodo programado.

Least- Connection

Este algoritmo dirige todo el flujo de paquetes hacia el servidor con menos carga dentro del cluster, para esto, rastrea las conexiones activas de todos los nodos y determina el servidor apropiado para recibir más peticiones. Es adecuado para grupos de servidores reales en donde cada nodo miembro tiene la misma capacidad.

Weighted Least-Connection

Al igual que el algoritmo de Least Connections, este elige al nodo basándose en sus conexiones activas, pero también basa su selección en la capacidad del servidor, este valor de capacidad, puede ser asignado a cada equipo de forma manual, los servidores con una mayor capacidad recibirán un mayor porcentaje de conexiones activas.

2.4.3 Flujo de paquetes en el balanceo

Durante una típica solicitud cliente/servidor hacia un servidor Web, el cliente en primer lugar establece una conexión TCP, luego envía una petición HTTP, recibe la respuesta y cierra la conexión TCP.

En el ejemplo siguiente existen 2 servidores Webs detrás de un balanceador: Cuando el balanceador recibe la solicitud TCP SYN, esta contiene la siguiente información:

- Dirección IP Origen. Esta es la dirección IP del cliente.
- Puerto Origen. Es el puerto usado por el cliente para establecer la conexión TCP.
- Dirección IP Destino. Es la dirección IP Virtual (VIP) que representa al cluster.
- Puerto Destino. Al ser un servidor Web las conexiones se realizan al puerto 80.

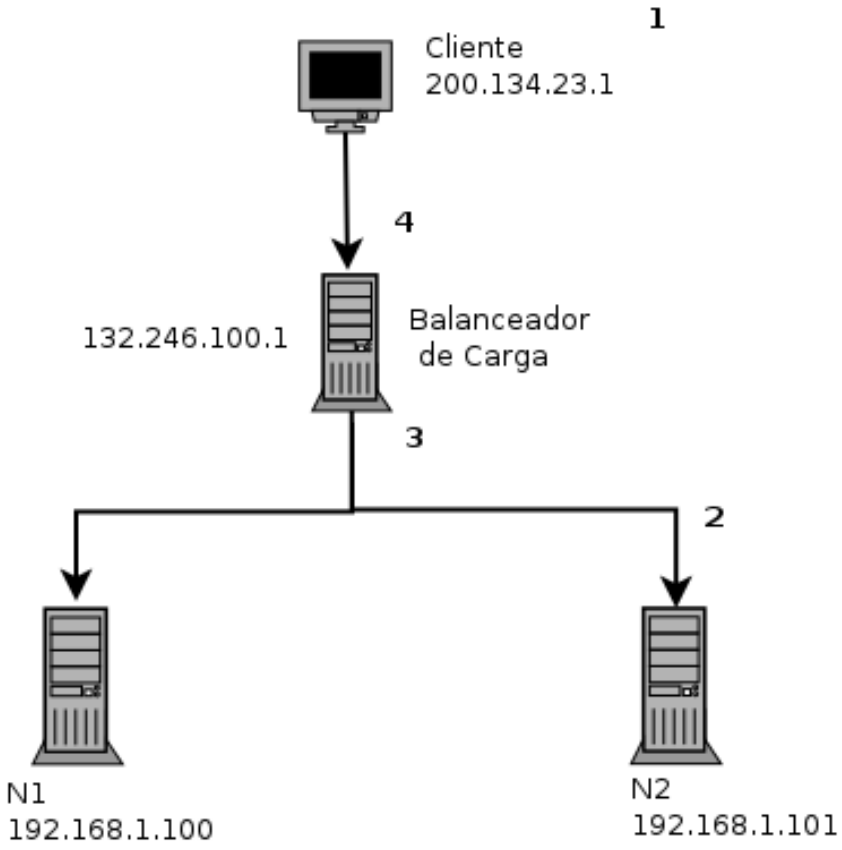

Figura 29: Ejemplo de flujo de paquetes.

Una vez que se recibe el primer paquete TCP, el balanceador comprueba en su tabla de sesiones si esta dirección IP y este puerto origen o puerto destino ya habían sido requeridos antes, si no encuentra ninguna entrada se da por hecho que se trata de una conexión nueva.

Después se elige a alguno de los nodos dependiendo de la configuración de balanceo y checks de salud, en este ejemplo las peticiones irán al servidor N2 para esta sesión, se crea una nueva entrada en la tabla de sesiones del balanceador y se procede a enviar el paquete al servidor N2.

Para que el servidor N2 pueda aceptar y procesar el paquete este debe de tener la dirección IP destino del mismo servidor, no la IP del balanceador (VIP), para lograr esto el balanceador cambia la VIP a la dirección IP del servidor N2 haciendo NAT, antes de reenviar el paquete.

El paquete llega al servidor N2, este lo recibe y envía su repuesta de vuelta al balanceador, el paquete de respuesta contiene la IP del servidor N2 por lo que el balanceador debe de realizar un proceso de un-NAT para remplazar la IP del servidor N2 por la VIP del balanceador, finalmente se responde a la petición al cliente.

Una vez que la conexión es terminada, el balanceador elimina la entrada en su tabla de sesión. En este ejemplo, todas los paquetes entrantes y salientes pasan por el balanceador, también existen otras técnicas de Balanceo de Carga para manipular las rutas de los paquetes según convenga, esto tomando en cuenta la arquitectura planeada, y el performance de las conexiones.

2.4.4 Técnicas de Balanceo de Carga IP

NAT

Esta técnica cambia la dirección de origen y destino de todos los paquetes por direcciones Ips definidas en el balanceador, antes de realizar el reenvió de paquetes al servidor real el balanceador cambia la dirección IP destino por la IP del servidor real.

Una vez que el servidor real recibe los paquetes, este ve a la VIP como dirección origen y responde las peticiones a esta. Cuando el balanceador recibe los paquetes este cambia la IP origen a la IP original del cliente y envía la respuesta.

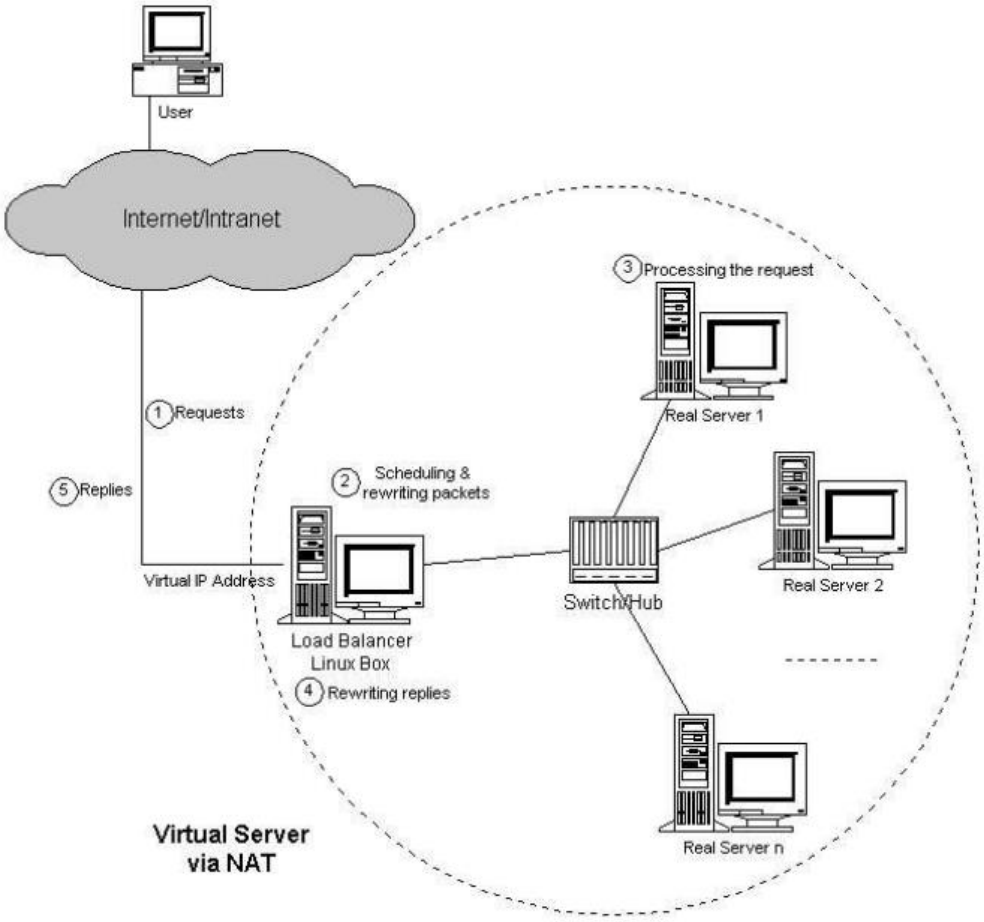

Figura 30: Balanceo IP por NAT.

Ruteo Directo

En esta técnica el tráfico de respuesta no pasa por le balanceador de carga, haciendo esto, es posible lograr una mejora en cuanto el rendimiento ya que se evita que el balanceador se convierta en un cuello de botella.

Para lograr esta técnica el balanceador no debe de cambiar la dirección IP que realiza la petición, para que la respuesta no requiera un UN-NAT y pueda así salir sin pasar por el balanceador.

Cuando se está bajo este tipo de técnica, el balanceador deja la dirección VIP como destino pero cambia la dirección MAC a la MAC del servidor real seleccionado según la política de balanceo. El servidor real acepta el paquete, una vez que el servidor responde la petición, la VIP pasa a ser la IP origen, esta pasa por el switch sin necesidad de pasar por el balanceador y llega al cliente.

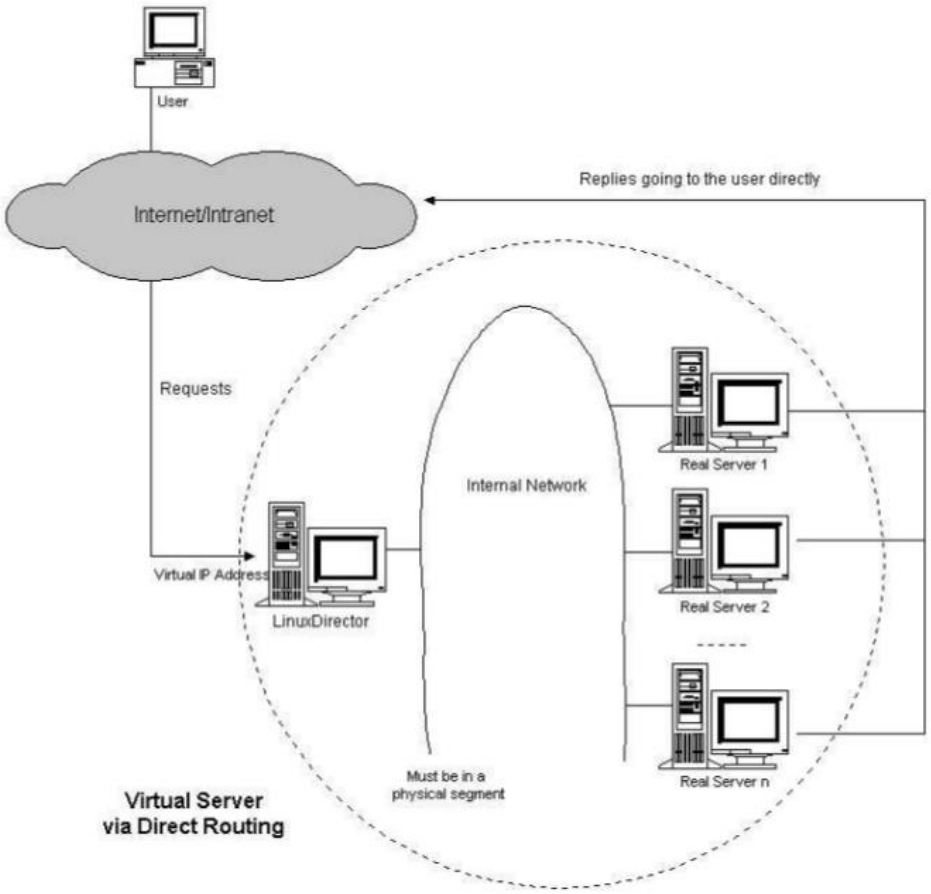

Figura 31: Balanceo IP por ruteo directo.

IP Tunneling

Esta técnica es usada para permitir que los servidores reales respondan desde distintas LANs o incluso WANs, ya que la comunicación se lleva a cabo mediante el protocolo de IP tuneling.

Este protocolo permite a un datagrama IP ser encapsulado dentro de otro datagrama, para lograr esto cada servidor real debe de soportar el protocolo de IP tunneling, además si el servidor real se encuentra en una red distinta que la del balanceador, el router de esa red necesita ser configurado para aceptar paquetes con la dirección IP virtual.

El flujo de paquetes en esta técnica es el siguiente:

- El Balanceador de cargas recibe el paquete del cliente lo encapsula dentro del datagrama IP y lo redirige dinámicamente a el servidor real.
- El servidor real recibe le paquete, lo desencapsula y busca dentro del paquete la dirección IP destino que concuerda con la IP virtual configurada en alguno de sus interfaces de túnel.
- El servidor real procesa la petición y es enviada directamente al usuario.

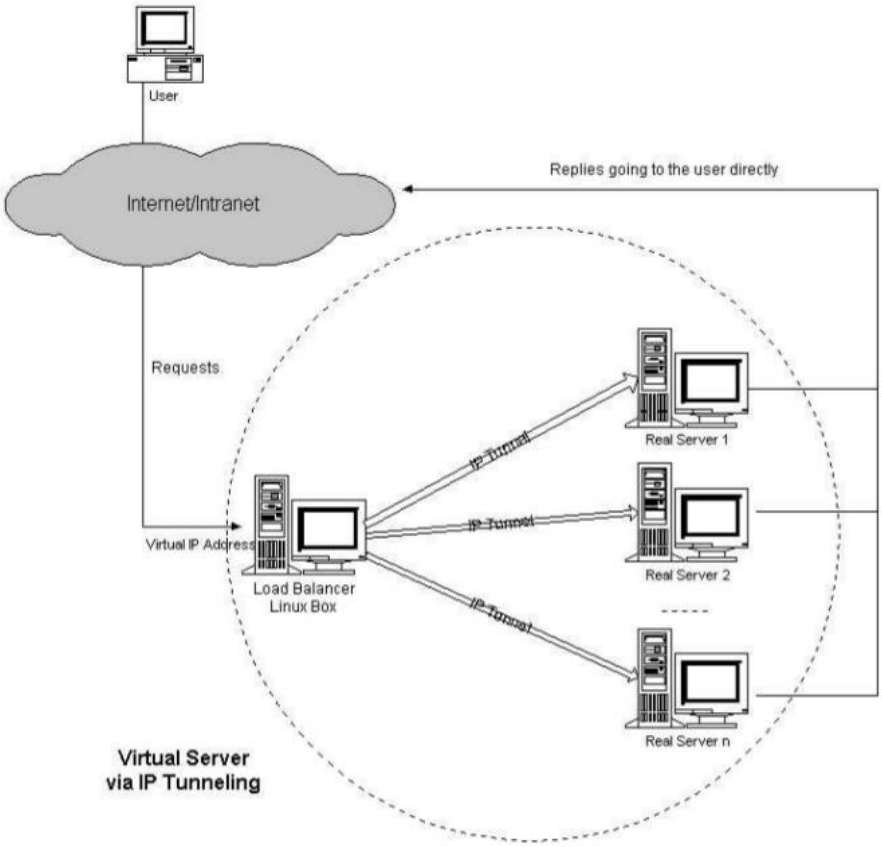

Figura 32: Balanceo IP por IP Tunneling.

2.4.5 Persistencia de sesión

Las aplicaciones web actuales utilizan sesiones para permitir a los usuarios una mayor interacción, identificar al usuario, y mostrar información personalizada al mismo, bajo un entorno de balanceo de carga el balanceador debe de enviar todas las conexiones realizadas por el usuario que inicio una sesión en la aplicación web al mismo servidor real durante todo el periodo de tiempo que esta exista o conservar los datos del usuario en caso de que el nodo en que se inició la conexión falle.

La persistencia de sesión de puede clasificar en 2 tipos:

Basada en información de paquetes TCP SYN

El balanceador recibe las peticiones de los clientes, éste revisa los datos que contienen los paquetes generados por las peticiones, particularmente la dirección IP origen del paquete, compara en su tabla de sesiones si esta dirección IP ya había realizado peticiones, si no existen en la tabla, trata a la conexión como nueva y la envía a un servidor real basándose en su política de balanceo, si ya existe una entrada con esta misma dirección IP dentro de la tabla de sesiones envía la petición al mismo servidor que la ocasión anterior, sin importar la política de balaceo.

Esta clase de persistencia de sesión no entiende los protocolos de aplicación, por lo cual no puede reconocer el momento en que una aplicación termina su sesión, para solucionar éste detalle, se establece un tiempo definido en que las IP‟s de las conexiones se mantendrán en la tabla de sesiones, una vez terminado éste periodo los registros se borraran.

Basada en información de la petición de aplicación

Éste tipo de persistencia funciona en el caso de aplicaciones Web, las peticiones HTTP contienen direcciones URLS y cookies que el balanceador puede considerar al momento de enviar la petición a un servidor real determinado.

Suponiendo que un cliente realiza una petición HTTP, al ser la primera conexión por parte del cliente no se tiene ninguna cookie, el balanceador envía la petición al servidor N2, el servidor N2 al darse cuenta que no existe ninguna cookie, crea una con el nombre de servidor y le asigna valor de 2, una vez que el cliente recibe la respuesta a su petición, la cookie se almacena en el disco duro del cliente. Cuando el cliente realiza una nueva conexión se envía la cookie servidor=2 como parte de la petición HTTP, el balanceador recibe la petición HTTP y lee el valor de la cookie, enviando la petición al servidor N2.

Hay que considerar en el diseño de la arquitectura que la sesión se perderá en caso que alguno de los servidores reales que almacena dichos datos falle, en este caso el balanceador enviará la petición al nodo que continué otorgando el servicio, pero la sesión se perderá ya que los datos no se encuentran en ese nodo.

Para poder solucionar éste problema es necesario almacenar los datos de sesión en algún servidor independiente o replicarlos entre todos los nodos del cluster, así cuando se presente una falla en algún nodo, los restantes mantendrán la sesión del usuario de forma transparente para este.

2.4.6 Soluciones para lograr alta disponibilidad y balanceo de carga

Existen varias soluciones para implementar una infraestructura de alta disponibilidad, éstas pueden ser productos de pago, que son desarrollados bajo un esquema de código cerrado y se instalan según las especificaciones del fabricante, o soluciones basadas en hardware, las cuales ofrecen un equipo activo extra que se instala dentro del cuarto de servidores y también contiene software propietario.

En ambos casos el precio es excesivamente alto y el nivel de personalización no suele ser muy bueno, ya que el usuario se debe de apegar a las características del producto y en caso de necesitar alguna configuración especial que su infraestructura demande es necesario un pago extra para poder ser desarrollado por parte del fabricante.

Por otro lado también existen soluciones de software libre que aunque no ofrecen ningún tipo de soporte o garantía, son extremadamente configurables y adaptables, con un precio prácticamente nulo, y debido a esto, suelen ser la elección obvia para pequeñas empresas que no cuentan con los recursos como para pagar soluciones propietarias. Sin embargo empresas grandes también hacen uso de ellas como complemento a sus sistemas propietarios.

A continuación se listan algunas soluciones libres para lograr alta disponibilidad y balanceo de carga en servicios y datos, es importante mencionar que en su conjunto y correctamente configurados ofrecen las mismas características que las opciones de pago:

Heartbeat

Es un software que permite la construcción de clusters de alta disponibilidad utilizando servidores comunes sin alguna dependencia específica de hardware. Es parte del proyecto Linux-HA (Linux High Availability) y es compatible con cualquier tipo de Linux y algunos Unixes como FreeBSD y Solaris.

Monitorea a los nodos donde es instalados enviado beats y en caso de no recibir respuesta levanta el servicio configurado para HA en el nodo restante.

Keepalived

Es un software de ruteo escrito en C su objetivo principal es el de proporcionar alta disponibilidad al proyecto LVS (Linux Virtual Server). KeepAlived es un servicio que monitoriza el estado del cluster LVS para que en caso de fallo el cluster siga en funcionamiento.

Pacemaker

Es un software de código abierto, su función es la de administrar los recursos dentro de un cluster soporta distintos escenarios, desde una configuración sencillas de solo 2 nodos, hasta 16 nodos con un esquema activo/activo.

Corosync

Es un grupo de sistemas de comunicación, con características especiales para implementar alta disponibilidad entre aplicaciones, trabaja como una capa de mensajes y permite a los administradores de recursos, comunicarse entre nodos y transferir los servicios.

DRBD

Es un software de replicación de datos, funciona simulando ser un RAID 1 a través de la red. Algunas de sus características son una replicación en tiempo real de manera continua transparencia en las aplicaciones que están almacenando datos en la unidad, posibilidad de recuperación de los datos ante un desastre.

MySQL Replication

Es una característica especial de MySQL la cual permite generar copias de las bases de datos y ser transferidas automáticamente de un nodo esclavo a uno maestro. Suele ser útil como medio respaldo, o análisis de datos cuando no se desea trabajar con la base de datos principal.

GlusterFS

Es un sistema de archivos que nos permite replicar y distribuir archivos a través de una red de comunicación, es una forma útil de crear una NAS con muy pocos recursos. Está diseñado para reducir la complejidad de un almacenamiento replicado, pero sin reducir el rendimiento y añade la característica de poder utilizar recursos sencillos para su implementación.
Dependiendo de la arquitectura en donde sea instalado, es posible lograr que la replicación de datos siga funcionando incluso si se presentaran fallas en los servidores donde se instaló el servicio.
Ofrece principalmente 2 tipos de volúmenes para el almacenamiento de datos:

- Replicado – Este método crea copias de los datos en todos los nodos, es la mejor opción para ambientes de alta disponibilidad.
- Distribuido – Este método almacena los archivos de forma distribuida en todos los nodos del cluster, se emplea para lograr un aumento en el rendimiento del IO

LVS (Linux Visrtual Servers)

Es una solución avanzada de balanceo de cargas que puede ser utilizada para construir una infraestructura escalable y con alta disponibilidad, multitud de servicios se pueden hacer uso de este software como: Web, cache, mail, ftp, media y servicios VoIP.

Mod_jk

Es un módulo del servidor Web apache que agrega la funcionalidad de balanceo de carga en las aplicaciones que se consultan, soporta distintos algoritmos y configuraciones, cabe señalar que está diseñado únicamente para balancear aplicación java.

Haproxy

HAProxy es un software que ofrece alta disponibilidad y balanceo de carga a nivel capa 7 del modelo OSI, está diseñado para trabajar con una alta carga de peticiones. Ofrece la posibilidad de configurar varios tipos de algoritmos de balanceo como: Raund Robin, Static Raund Robin, Least Connection, Source, URL.
Debido a su funcionamiento en capa 7 del modelo OSI, es posible interactuar con el aplicativo de forma más inteligente que otros balanceadores.

Memcache

Es un sistema de cache distribuido, tienen usos diversos pero principalmente se usa para aumentar la velocidad de acceso en una web dinámica, aunque también se puede usar para dar alta disponibilidad a las sesiones de usuario.

Galera MariaDB

Es un servicio de cluster síncrono y con esquema master-master para el servidor de base de datos MariaBD, está disponible sólo para Linux y soporta los motores de base de datos XtraDB/InnoDB.

Red Hat cluster suite

Es un conjunto de servicios que brindan alta disponibilidad, se configura por medio de una interfaz web y está disponible desde los repositorios oficiales de Red Hat, es necesario comprar una suscripción para contar con soporte y actualizaciones oficiales.

2.5 Problemas en Clusters

Existen varios problemas que se presentan en las arquitecturas con clusters, los cuales generalmente son fallas en la sincronización de los datos o en servicios realojados de forma errónea en alguno de los nodos que componen al cluster, esto provoca que el aplicativo al que se le está dado alta disponibilidad responda a peticiones de manera incorrecta o se vuelva inaccesible.

Otro tipo más grave de problema que suele presentarse es el Split Brain, éste tipo de error se presenta en mayor medida en clusters de dos nodos, en este escenario el nodo A falla y el nodo B toma el control, el aplicativo continua trabajando y escribiendo datos en el nodo B, una vez que el nodo A realiza su proceso de failback el software encargado de realojar los recursos no sabe cuál de los nodos tiene los datos actualizados por lo que toma el nodo A como nodo principal siendo que no cuenta con la información actualizada causando corrupción de datos.

Existen métodos eficaces para evitar el Splitbrain, es necesario asegurar que el nodo que falla no sea tomado en cuenta por el administrador de recursos hasta que cuente con la información actualizada, y en caso que esto no sea posible realizar un proceso de Fencing en el.

2.6 Pruebas al Cluster

La fase final en la implementación de un cluster par alta disponibilidad, es el periodo de pruebas, en él es necesario revisar uno a uno que los componentes del cluster cuenten con una correcta redundancia. En esta etapa se realizan pruebas de falla a nivel hardware y software, de miden los tiempos de respuesta y son detectados errores en la configuración.

Para estas pruebas es común el uso de benchmarks con el fin de estresar al cluster y observar su comportamiento. Una vez concluido el periodo de pruebas es posible implementar el cluster en un ambiente productivo con su adecuado monitoreo.

CAPÍTULO III CONFIGURACIÓN E IMPLEMENTACIÓN

3.1 Propuesta de solución

Se propone el diseño e implementación de un cluster de alta disponibilidad y balanceo de carga, el cual albergara el aplicativo de aulas virtuales Moodle, y le otorgará a éste último la posibilidad de continuar respondiendo a peticiones aun cuando se presenten fallas en el sistema. Moodle es una aplicación web, escrita en el lenguaje de programación PHP, requiere de un servidor web, un servidor de base de datos, así como de un espacio disponible para guardar archivos que después se puedan consultar. A continuación se identifican las partes de la aplicación a las que se les requiere otorgar HA:

Servicio web.- Otorgar alta disponibilidad al servicio, recuperar la sesión de usuario, balancear peticiones recibidas en estos nodos.

Base de datos.- Alta disponibilidad de datos, asegurar integridad de la información, contar con un balanceo de peticiones hacia la base de datos.

Archivos generados.- Replicar archivos para asegurar su integridad

Después de observar el comportamiento del aplicativo Moodle, sus necesidades, así como identificar el flujo básico del programa y las áreas críticas en donde se requiere contar con HA , se estableció que el uso del siguiente software es el adecuado para construir la solución:

- Heartbeat, cuenta con mucha documentación además de ser un proyecto estable y maduro, es posible su integración a casi cualquier programa debido a la facilidad de personalización de sus scripts.
- Haproxy, fue utilizado debido a su integración con el aplicativo Web y la posibilidad de realizar checks de funcionamiento, además es capaz de integrar a los nodos de base de datos a su pool de balanceo.

- GlusterFS, realiza la replicación entre nodos de forma sencilla y en alta disponibilidad que se requiere para que Moodle conserve la sesión y archivos generados, esto lo realiza sin modificación alguna en la configuración del aplicativo, lo que hace que sea versátil y sencillo de modificar.
- MariaBD, es compatible con Moodle y cuenta con características más avanzadas de replicación que MySQL, además su complemento Galera ofrece características de alta calidad para el balanceo y alta disponibilidad del servicio, en pruebas no se presentaron errores de SplitBrain durante procesos de Faillover.

Dentro de la arquitectura propuesta existen servidores en estado activo/activo (nodos de base de datos y webs) así como en activo/pasivo (balanceadores).

La técnica de balanceo utilizada será de tipo NAT, con persistencia de sesión basada en información de la petición de aplicación, en caso de falla la sesión es replicada a los demás nodos

3.2 Requerimientos de infraestructura

Para construir el cluster se utilizaron un total de 6 PCs y durante las pruebas de agregaron 2 más, todos los nodos forman parte del laboratorio de cómputo de la licenciatura en informática, lugar en donde fue realizado el proyecto.

La arquitectura principal consta de 6 nodos, divididos en 3 capas:

- Capa de balanceo.
- Capa web.
- Capa de base de datos.

Capa de Balanceo

Consiste de 2 nodos, uno activo y el segundo pasivo, cada nodo cuenta con tres adaptadores de red.

- El primer adaptador estará conectado al switch de la DMZ.
- El segundo adaptador será un cable cruzado entre los nodos.
- El tercer adaptador estará conectado al switch de la LAN.

Esta configuración deberá ser la misma en ambos nodos, ya que si el balanceador activo falla, el nodo pasivo no deberá tener problemas para seguir proporcionando el servicio.

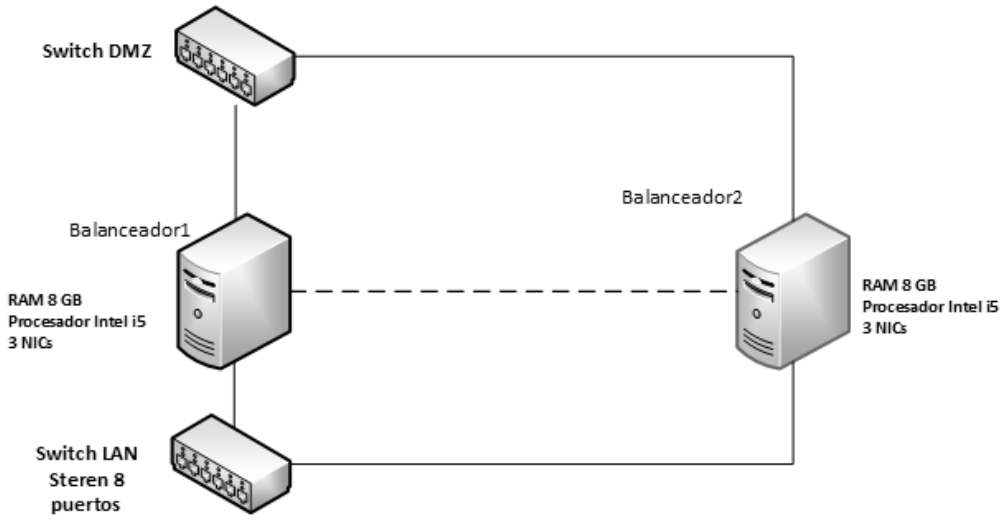

Figura 33: Componentes de Capa de Balanceo

Capa web

Ésta capa consiste de 2 nodos conectados al switch de la LAN y una interfaz de red por cada nodo.
La capa web es escalable, ya que se pueden agregar cuantos nodos sean necesarios, tomando en cuenta los puertos disponibles del switch. Durante las pruebas se escalara a 4 nodos.

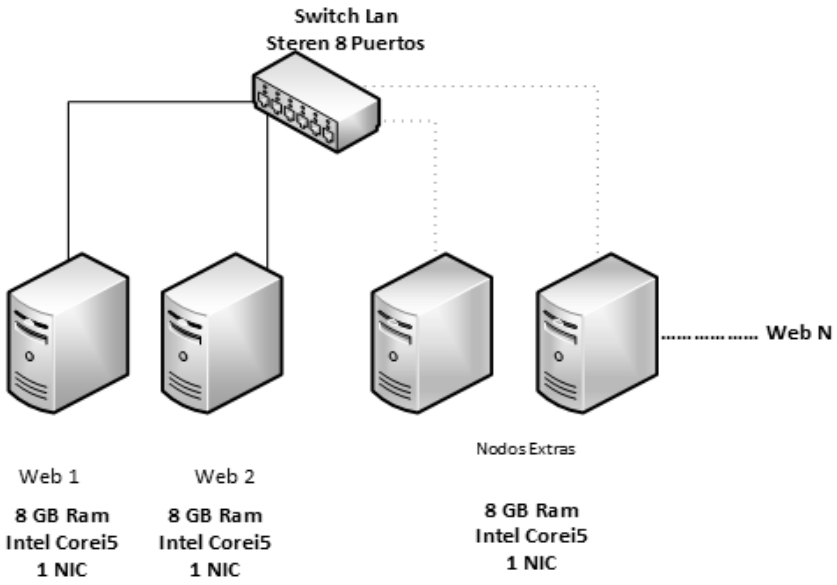

Figura 34: Componentes de Capa Web

Capa de base de datos

Ésta capa, está compuesta de dos nodos de base de datos, en esquema activo/activo, y de igual forma que la capa web puede ser escalable. En ambos nodos el adaptador está conectado al switch de la LAN.

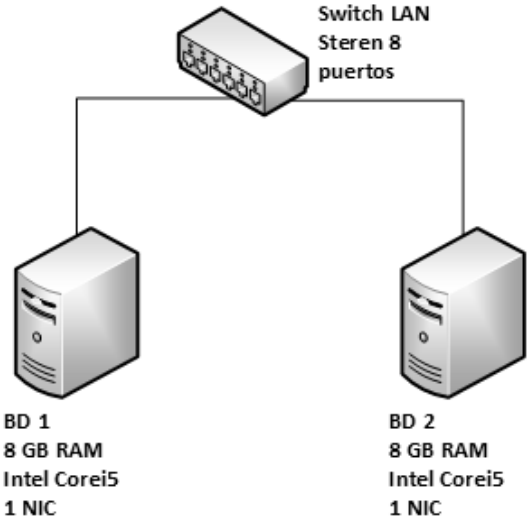

Figura 35: Componentes de Capa Base de Datos

Para completar la arquitectura, además de los equipos antes mencionados, se utilizó un switch de ocho puertos para interconectar las capas, cables UTP, uno de ellos cruzado y 4 adaptadores

de red extras, los cuales fueron insertados en los dos balanceadores. El esquema completo se muestra en la siguiente figura:

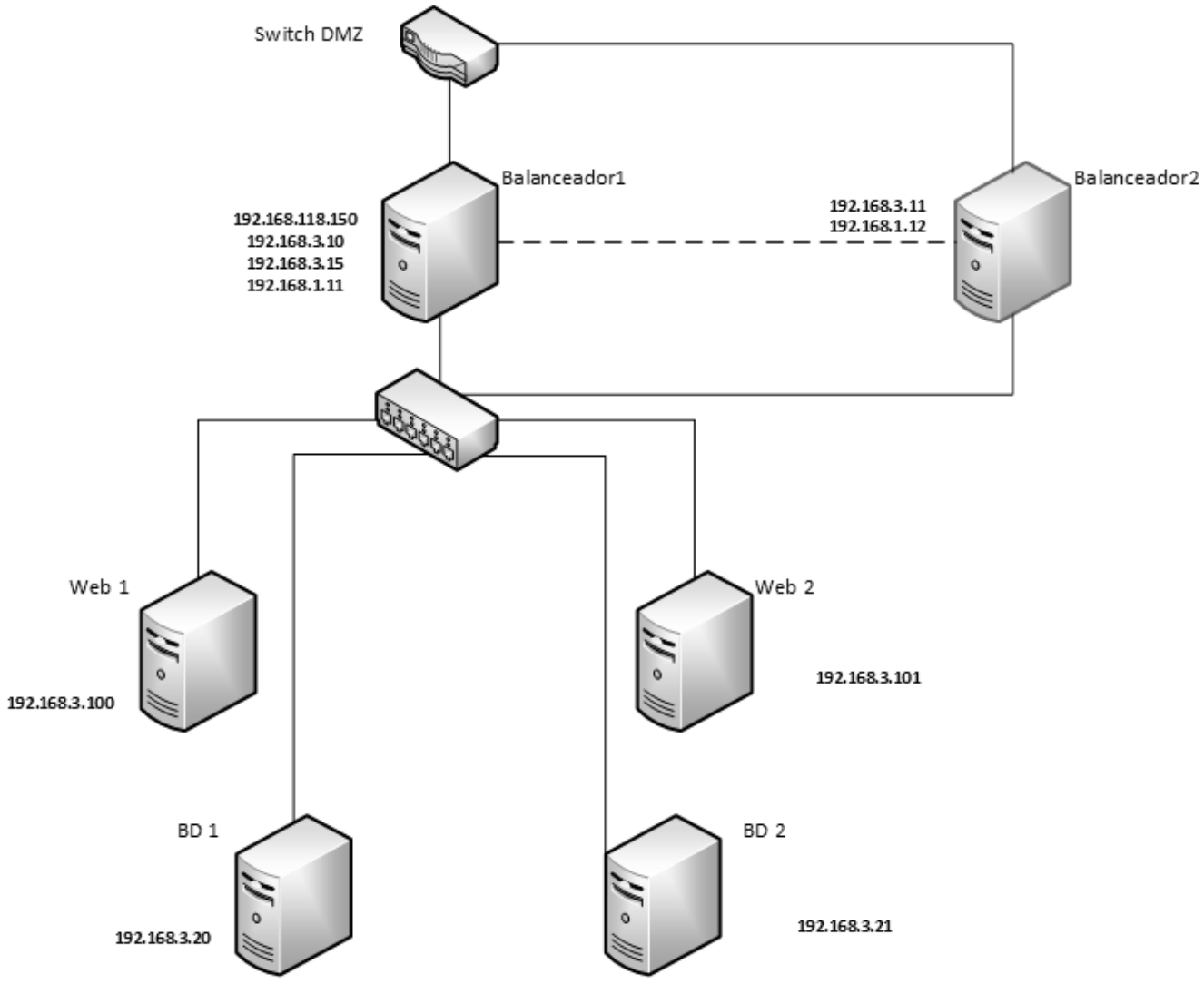

Figura 36: Arquitectura Completa

3.3 Implementación

Al cluster se le instalaron diversos tipos de software especializados, todos ellos son de licencia libre. De igual manera se clasificaron software de balanceo, software de aplicación web y software de base de datos como lo muestra la *figura 37*

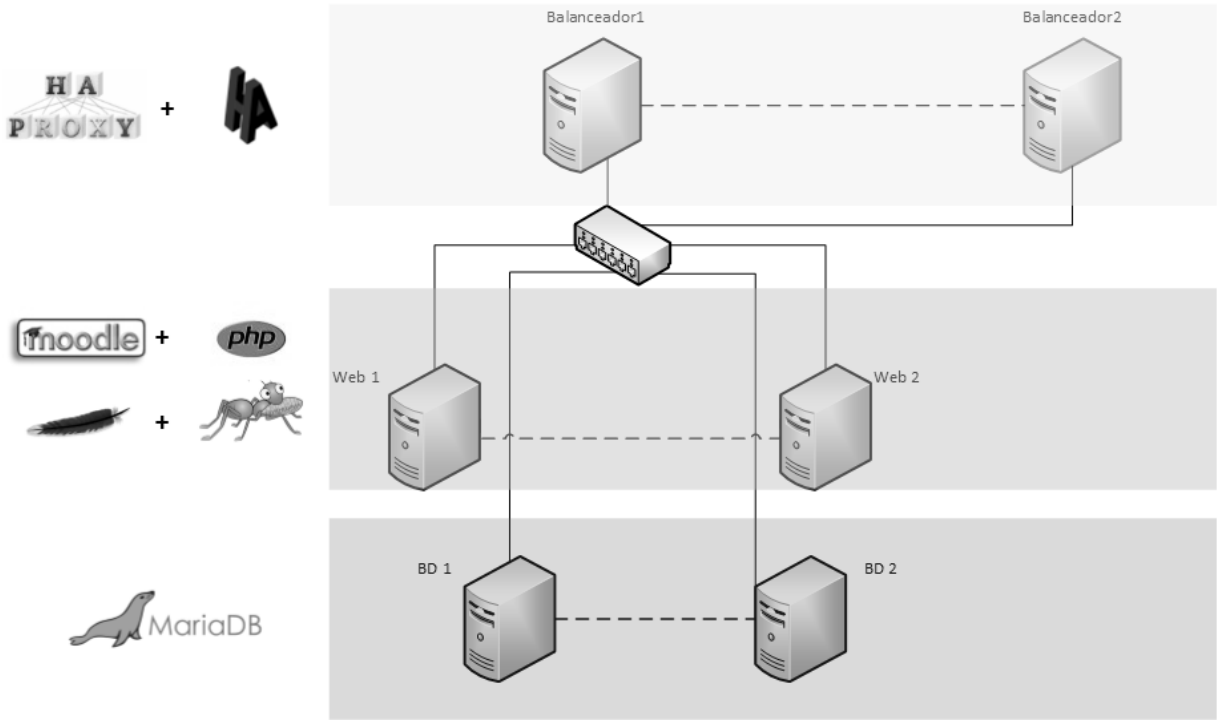

Figura 37: Esquema de Software

3.3.1 Plan de implementación

Para la realización de este proyecto se usó como software base el sistema operativo Linux Centos 6.5 con las últimas actualizaciones instaladas y sin ninguna modificación extra en el sistema. El motivo de usar esta distribución Linux es su amplia documentación disponible en internet, así como su amplio uso en servidores productivos en el mundo. Se realizó la

instalación del software para el Cluster HA+LB, en el siguiente orden:

1. Instalar y configurar MariaBD Galera Cluster en los dos nodos de la capa de Base de Datos.
2. Instalar y configurar Apache y GlusterFS en los dos nodos de la capa Web.
3. Instalar y configurar Haproxy y Heartbeat en los dos nodos de la capa de balanceo.
4. Por último se instalará el aplicativo Moodle realizando las configuraciones necesarias para su correcto funcionamiento.

En el apartado de anexos se encuentran diversos scripts que se desarrollaron para automatizar la instalación de futuras implementaciones.

3.3.2 Capa de Base de Datos

A los nodos que componen la capa de base de datos se les fue asignada una IP del segmento de la LAN.

Nodo1	192.168.3.20	Base2-lan
Nodo2	192.168.3.21	Base2-lan

Tabla 4: IPs para capa de base de datos

Ésta capa consiste en dos nodos a los cuales se les fue instalado MariaBD Galera Cluster, ambos se encuentran configurados en un esquema activo/activo y se replican de forma síncrona.

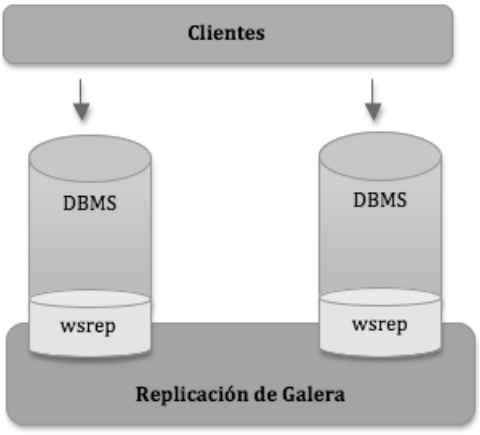

Figura 38: Galera MariaBD Cluster

Instalación de Galera MariaBD

Se inició con la instalación agregando el repositorio de MariaBD en ambos nodos de base de datos, una vez cargado el repositorio, se instalaron los paquetes necesarios.

```
/etc/yum.repos.d/maria.repo

[mariadb]

name = MariaDB

baseurl = http://yum.mariadb.org/5.5/centos6-amd64

gpgkey=https://yum.mariadb.org/RPM-GPG-KEY-MariaDB

gpgcheck=1

yum install MariaDB-Galera-server MariaDB-client galera

yum install http://www.percona.com/downloads/percona-release/redhat/0.1-3/percona-release-0.1-3.noarch.rpm

yum install percona-toolkit percona-xtrabackup
```

Después de haber terminado el proceso de instalación de los paquetes, se procede con la configuración del cluster, solo en el nodo 1 de base, ejecutando los siguientes comandos:

```
[root@base1 ~]# service mysql start

 [root@base1 ~]# mysql_secure_installation

Enter current password for root (enter for none):

OK, successfully used password, moving on...

Set root password? [Y/n] y

New password:           Password updated successfully!

Reloading privilege tables..   ... Success!

Remove anonymous users? [Y/n] n  ... skipping.

Disallow root login remotely? [Y/n] n  ... skipping.

Remove test database and access to it? [Y/n]   - Dropping test database...... Success!

 - Removing privileges on test database...  ... Success!

Reload privilege tables now? [Y/n]   ... Success!
```

Luego, es necesario configurar cada uno de los nodos, el archivo de configuración del cluster se encuentra en la ruta /etc/my.cnf.d/server.cnf, aquí se especifican los siguientes parámetros:

wsrep_provider	Se especifica la librería de Galera
wsrep_cluster_address	En esta parte se incluyen las direcciones IPs de los nodos de Base de Datos
wsrep_cluster_name	Se especifica el nombre del cluster de Base de Datos
wsrep_node_address	IP del nodo donde se configura el archivo
wsrep_node_name	Hostname del servidor
wsrep_sst_auth	Usuario y password para autentificar la réplica de datos

Tabla 5: Parámetros Galera

Las configuraciones utilizadas para ambos nodos del cluster galera se muestran en la siguiente tabla:

Nodo 1	**Nodo 2**
[mariadb] query_cache_size=0 binlog_format=ROW default_storage_engine=innodb innodb_autoinc_lock_mode=2 wsrep_provider=/usr/lib64/galera/libgalera_smm.so wsrep_cluster_address=gcomm://192.168.3.21 wsrep_cluster_name=clusterbd wsrep_node_address=192.168.3.20 wsrep_node_name=base1 wsrep_sst_method=rsync wsrep_sst_auth=root:tw1nn0r1tt@	[mariadb] query_cache_size=0 binlog_format=ROW default_storage_engine=innodb innodb_autoinc_lock_mode=2 wsrep_provider=/usr/lib64/galera/libgalera_smm.so wsrep_cluster_address=gcomm://192.168.3.20 wsrep_cluster_name=clusterbd wsrep_node_address=192.168.3.21 wsrep_node_name=base2 wsrep_sst_method=rsync wsrep_sst_auth=root:tw1nn0r1tt@

Tabla 6: Configuraciones de nodos de base de datos

Iniciar Maria BD con los siguientes comandos:

```
En el nodo 1:

/etc/init.d/mysql stop

/etc/init.d/mysql bootstrap

En el nodo 2:

/etc/init.d/mysql start
```

Con ésta configuración ya se tiene lista la replicación de datos entre los dos nodos, se continúa con la configuración necesaria para integrar estos dos nodos con la capa de balanceo.

Para la integración de la capa de base de datos con la capa de balanceo es necesario el uso de un script de revisión de servicio, este script monitorea cada nodo de MariaBD regresando el valor 200 si el nodo se encuentra activo o 503 si existe algún problema con el nodo, Haproxy utiliza estos valores para determinar hacia que nodo envía las peticiones por parte del cliente.

En ambos nodos se copia el script en la ruta /usr/bin. Aquí se deben de configurar un usuario y contraseña el cual se conectará a la base de datos, se editan los siguientes valores dentro del script:

```
MYSQL_USERNAME="${1-usuario}"

MYSQL_PASSWORD="${2-pass}"
```

A continuación es necesario agregar un usuario de prueba en la base de datos, para que el script de chequeo se conecte a la base de datos, el nombre de usuario y password son los mismos que se definen en el script anterior

```
[root@base1 ~]# mysql -u root -p

MariaDB [(none)]> GRANT PROCESS ON *.* TO 'usuario'@'localhost'
IDENTIFIED BY 'pass' ;
```

Siguiendo con la integración a la capa de balanceo, se instala el paquete xinetd en ambos nodos:

```
Yum install xinetd
```

Xinietd es un demonio que puede iniciar otros servicios cuando una petición llega al puerto especificado en su archivo de configuración, se estableció la configuración en la ruta /etc/xinetd.d/mysqlchk para ambos nodos de base de datos de la siguiente manera:

```
disable = no
flags = REUSE
socket_type = stream
port = 9200
wait = no
user = nobody
server = /usr/bin/clustercheck
log_on_failure += USERID
only_from = 0.0.0.0/0
per_sourse = UNLIMITED
```

También es necesario declarar el puerto 9200 que es por donde estará escuchando el servicio dentro del archivo /etc/services.

Por último se inicia el servicio en ambos nodos:

```
# service xinetd start

Starting xinetd:
[  OK  ]
```

3.3.3 Capa web

La capa web de la arquitectura cuenta con 2 nodos, a cada uno se le asignó una dirección IP del segmento de la LAN

Nodo1	192.168.3.100	web1
Nodo2	192.168.3.101	web2

Tabla 7: IPs de capa web

Cada uno de ellos tiene instalado un sistema Linux CentOS 6.5 y les fueron cargados los

paquetes necesarios para el funcionamiento de la plataforma Moodle.
Se instaló el servidor web apache y los paquetes necesarios para ejecutar aplicaciones de PHP, además de paquetes necesarios para el correcto funcionamiento del aplicativo Moodle.

```
yum install httpd

yum install php  php-mysql php-pdo php-gd php-soap php-xmlrpc php-xml  php-
cli php-mbstring php-pear mod_ssl
```

Con la instalación de estos paquetes se tiene el esquema base de un servidor web con soporte para Moodle, lo siguiente es instalar y configurar lo nodos para alta disponibilidad.

Instalación de GlusterFS

Moodle almacena, además de sesiones de acceso y cookies, archivos cargados por los usuarios. Estos son guardados en una configuración por default en el directorio /moodledata, por lo que es necesario también replicar estos datos hacia todos los nodos, asegurándonos así que en caso de que un nodo falle, los nodos restantes tuvieran los datos de éste, logrando con esto persistencia a nivel sesión de usuario y a nivel datos. Para éste fin es usará el sistema de archivos GlusterFS

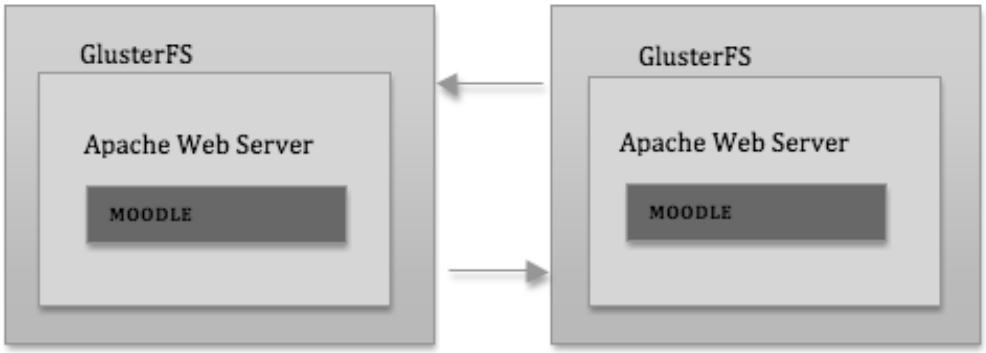

Figura 39: Alta disponibilidad en Apache usando GlusterFS

Es necesario instalar el repositorio que contiene los paquetes de GlusterFS ya que éste

software no se encuentra en los repositorios oficiales de la distribución. En el esquema propuesto de 2 nodos, ambos juegan el papel de cliente y de servidor a la vez, por lo que se procede también a instalar en ambos nodos los paquetes de cliente y de servidor.

```
wget -P /etc/yum.repos.d
http://download.gluster.org/pub/gluster/glusterfs/LATEST/CentOS/gluster
fs-epel.repo

yum -y install glusterfs glusterfs-fuse glusterfs-server glusterfs-
client
```

Se crea un directorio en ambos nodos que será utilizado para montar el sistema de archivos de GlusterFS. Luego se crea el nuevo volumen especificando nombre y el tipo de volumen, en este caso elegimos el de replicación por transporte TCP.

```
mkdir  /mnt/datos

[root@web01 glusterfs]# gluster volume create datos replica 2
transport tcp web1:/datos web2:/datos force

volume create: datos: success: please start the volume to access
data
```

Se inicia el sistema de Archivos y el mapeo del nuevo sistema de archivos en ambos nodos

```
[root@web01 glusterfs]# gluster volume start datos

volume start: datos: success

mount -t glusterfs -oattribute-timeout=1,entry-timeout=1
web1:/datos /mnt/datos

mount -t glusterfs -oattribute-timeout=1,entry-timeout=1
web2:/datos /mnt/datos
```

Ahora ambos nodos ya tienen montados el sistema de archivos y se encuentran replicando entre ellos los datos generados por Moodle, entre ellos la sesión de usuario y los datos que el usuario pueda cargar en el aplicativo. Con esta configuración, si alguno de los nodos Web falla, la información estará disponible en los equipos que continúen contestando a peticiones. Una vez que el nodo afectado regrese al cluster, los datos que hayan sido escritos durante el tiempo en que no estuvo disponible serán replicados hacia él, desde los demás nodos. Dado que Moodle vive en este Sistema de Archivos, el balanceador no enviara peticiones al nodo afectado hasta que el aplicativo se encuentre disponible y totalmente actualizado.

3.3.4 Capa de balanceo

La capa de balanceo está formada por dos nodos a los que se les configuro su respectiva IP del segmento LAN y comparten dos VIPs una correspondiente a la LAN y la segunda otorgada por la red pública del segmento del laboratorio de informática.

Nodo 1	192.168.3.10, 192.168.1.12	Balanceador-1
Nodo 2	192.168.3.11, 192.168.1.12	Balanceador-2
VIP: 192.168.118.150 VIP-LAN 192.168.3.15		

Tabla 8: IPs para capa de balanceo

Sus Interfaces de red fueron configuradas en base a la arquitectura propuesta:

- Eth0.- Ip pública.
- Eth1.- Interfaz utilizada para el envío de mensajes heartbeats entre los dos servidores.
- Eth2.- Interfaz utilizada para comunicarse con la LAN.

Los 2 nodos usados como balanceo de carga fueron instalados con el esquema por default sin alguna configuración especial en el sistema operativo.

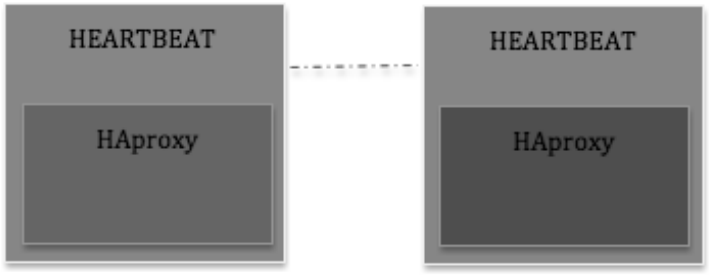

Figura 40: Nodo Pasivo y Nodo Activo Heartbeat y HAproxy

A estos dos nodos les fue instalado el software Heartbeat y sobre de éste se instaló el balanceador de carga HAproxy.

Instalación y configuración de Software de Balanceo

Se instaló haproxy desde los repositorios oficiales de CentOS Linux, la versión instalada fue la 1.4.24-2.el6

```
yum install haproxy
```

Se realizaron dos configuraciones en el balanceador, en el primer pool se establece el balanceo para los servidores Web, con dos algoritmos distintos Round Robin y Least Conection.

Para los servidores de base de datos se creó otro pool de balanceo, estableciendo dos políticas en la configuración, Round Robin y Failover con persistencia de nodo. Las configuraciones del balanceador se establecen en el archivo /etc/haproxy/haproxy.cfg

Pool de Servidores Web

Estableciendo el valor “roundrobin” en el parámetro balance y declarando las Ips o hostnames en las líneas del pool de balanceo, haproxy redirecionará todas las peticiones que se realizan a la VIP 192.168.118.150 hacia los servidores web, balanceando la carga uno a uno.

```
backend servers
      balance roundrobin
      option httpchk HEAD/HTTP/1.0
      cookie SERVERID insert indirect nocache
      server s1 web1:80 check cookie s1
      server s1 web2:80 check cookie s2
```

Configuración Balanceando por Round-Robin haproxy.cfg

Además la línea option httpchk, realiza una revisión del estado del servidor Web, enviando una petición HEAD, si el servidor Web responde con un error o no responde, el Server correspondiente no será considerado para peticiones futuras. Esto ayuda a detectar posibles errores a nivel aplicación.

El segundo método de balanceo utilizado fue el de Least Connection, estableciendo el valor "leastconn" en el parámetro balance, haproxy comienza a balancear tomando en cuenta el número de conexiones activas de cada nodo y enviando peticiones al nodo de la capa web que tenga menos conexiones activas.

```
backend servers
      balance leastconn
      option httpchk HEAD/HTTP/1.0
      cookie SERVERID insert indirect nocache
      server s1 web1:80 check cookie s1
      server s1 web2:80 check cookie s2
```

Configuración: Balanceo por LeastConnection

Pool de servidores MariaBD

La primera política establecida dentro de este pool (la cual está escuchando por el puerto 13305) no realiza ningún balanceo entre nodos, todas las peticiones siempre son dirigidas hacia el mismo nodo, aun cuando ambos están en modo activo/activo. En caso de falla en el nodo preferido de forma inmediata y sin downtime alguno, el servidor secundario continuara sirviendo a las peticiones.

```
mode tcp
option httpchk
server galera-node1 192.168.3.20:3306 check port 9200
server galera-node2 912.168.3.21:3306 check port 9200
```

Configuración failover para nodos de Base de Datos

La segunda configuración dentro de este Pool (la cual está escuchando por el puerto 13304 y es la que se usara en las pruebas), establece un balanceo con un algoritmo Least Connection Failover, en ésta ocasión haproxy enviara las peticiones al nodo con menos conexiones activas uno a uno y en caso de fallar las enviará al nodo que esté operando de manera correcta.

```
mode tcp
balance leastconn
option httpchk
server galera-node1 192.168.3.20:3306 check port 9200
server galera-node2 912.168.3.21:3306 check port 9200
```

Configuración Least connection para nodos de Base de Datos

Una vez concluida con la configuración de HaProxy en ambos nodos, se procede a la instalación de HeartBeat en los balanceadores. Es importante tener especial atención en la configuración del demonio de HeartBeat ya que de lo contrario podría provocar problemas de Split Brain en el cluster.

El paquete de Heartbeat no se encuentra incluido en los repositorios oficiales de CentOS Linux, por lo que se agregaron los repositorios EPEL y remi-release para poder instalar el paquete. Ya con los repositorios instalados y activados, se procedió a instalar los paquetes de heartbeat y heartbeat-libs desde el administrador de paquetes yum.

```
rpm -Uvh http://download.fedoraproject.org/pub/epel/6/x86_64/epel-
release-6-8.noarch.rpm
yum install heartbeat
```

Una vez concluida la instalación del software es posible continuar con la configuración de acuerdo a la arquitectura planteada para el cluster.

Configuración de Heartbeat

El archivo de configuración principal de heartbeat se encuentra dentro del directorio /etc/ha.d/ y tiene como nombre ha.cf, en este archivo se establecen parámetros como: tiempo de espera antes de considerar a un nodo no disponible, nombre de las interfaces de red por donde serán enviados los beats, hostname de los nodos.

A continuación se muestra el archivo de configuración del balanceador activo y pasivo:

Balanceador 1 (Activo)	**Balanceador 2 (Pasivo)**
logfile /var/log/ha.log logfacility local0 keepalive 2 deadtime 5 initdead 60 bcast eth1 ucast eth1 192.168.1.12 udp eth1 udpport 694 auto_failback on node balanceador-1 node balanceador-2	logfile /var/log/ha.log logfacility local0 keepalive 2 deadtime 5 initdead 60 bcast eth1 ucast eth1 192.168.1.11 udp eth1 udpport 694 auto_failback on node balanceador-1 node balanceador-2

Tabla 9: Configuraciones en capa de balanceo

El segundo archivo de configuración que es necesario editar es /etc/ha.d/haresources aquí se establecen las acciones que heartbeat realizará al momento de detectar falla en alguno de los nodos. A continuación se muestra el contenido del archivo de configuración para ambos nodos.

Balanceador Activo	**Balanceador Pasivo**
balanceador-1 IPaddr::192.168.118.150/24/eth0 balanceador-1 IPaddr::192.168.3.15/24/eth2	balanceador-2 IPaddr::192.168.118.150/24/eth0 balanceador-2 IPaddr::192.168.3.15/24/eth2

Tabla 10: Configuración Heartbeat

Como se observa en la tabla anterior, se establece el hostname del respectivo nodo, seguido de

la dirección virtual, estas dos líneas de configuración provocarán que al nodo se le asigne la VIP 192.168.118.150 y la VIP-LAN 192.168.3.15 si su par falla.

Es necesario crear el archivo /etc/ha.d/authkeys en ambos nodos, en el cual se especificará el método de autentificación que usa heartbeat para integrar a los nodos dentro del cluster. Una vez terminada la configuración se puede iniciar heartbeat.

```
auth 3
3 md5 passfesc-4
chmod 600 /etc/ha.d/authkeys

/etc/init.d/heartbeat restart
```

3.3.5 Instalación y configuración del Aplicativo Moodle

Se instaló el aplicativo Moodle, siguiendo el manual de instalación de su web oficial (https://www.moodle.org), cabe mencionar que a nivel aplicación no se realizó ningún tunning o modificación, sólo fue necesario realizar pequeñas modificaciones para integrar la plataforma a la arquitectura HA, las cuales se detallan a continuación:

Seleccionar la VIP-LAN 192.168.3.15 correspondiente a la capa de balanceo para la configuración de la Base de Datos, esto hará que el aplicativo envié las peticiones al balanceador.

Figura 41: Configuración de Base de Datos de Moodle

De igual forma, en el archivo config.php es necesario definir la dirección VIP de la capa de balanceo y el directorio de moodledata el cual se encuentra creado sobre el sistema de archivos de GlusterFS.

```
$CFG->wwwroot   = 'http://192.168.1.150/moodle';

$CFG->dataroot  = '/datos/moodledata';

$CFG->admin     = 'admin';
```

Configuración de config.php

3.4 Flujo de datos en el cluster

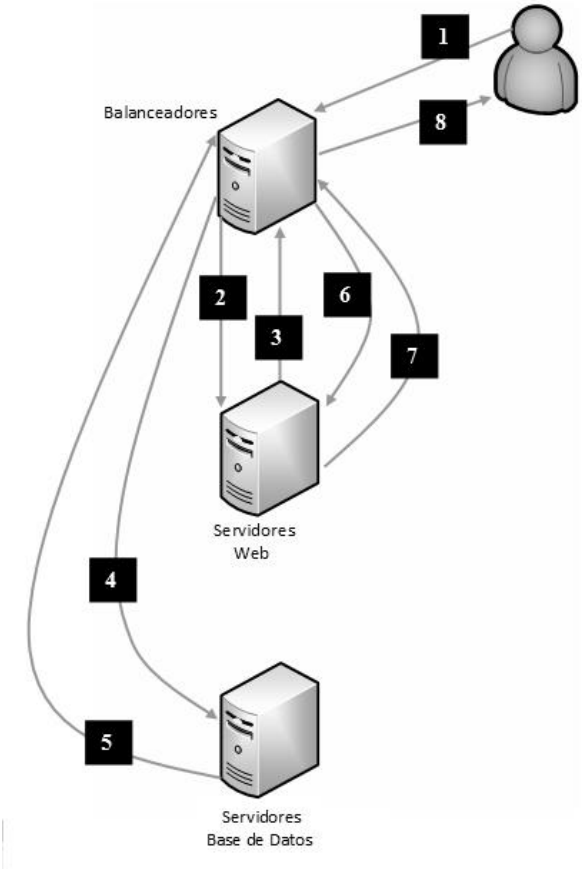

Figura 42: Flujo de Paquete dentro del Cluster

El flujo completo de peticiones dentro de la arquitectura propuesta es realizado en 8 Pasos.

Comenzando por una consulta a Moodle enviada desde el Cliente **(1),** ésta llega al HAproxy por el puerto 80, el cual la envía a alguno de los nodos Web **(2)** éste nodo consulta la VIP de base de datos la cual pertenece a Haproxy **(3)**, el balanceador envía la consulta a alguno de los 2 nodos de Galera MariaBD **(4)**, el nodo de base de datos contesta a la consulta al Haproxy **(5)**, luego el balanceador entrega el resultado al nodo web que lo solicito **(6)**, el nodo web envía la petición al balanceador **(7)**, finalmente el balanceador entrega la respuesta al cliente **(8)**.Las peticiones son enviadas a los diferentes servidores reales dependiendo de la carga y la disponibilidad que tengan los nodos.

3.5 Escalamiento de Capa Web

Dentro del diseño del Cluster se estableció que la capa Web seria escalable dependiendo de la carga que presente el aplicativo, el proceso de escalamiento consta de 3 pasos principales:

1. Replicar la configuración de los nodos Web, asignándole al nuevo servidor una IP del mismo segmento.

2. Reconfigurar GlusterFS para que replique el FS de Moodle en el nuevo nodo de la capa Web, montar el sistema se archivos compartidos en el nuevo nodo e iniciar servidor apache.

```
 [root@web01 glusterfs]# gluster volume create datos replica 2
transport tcp web1:/datos web2:/datos web3:/datos force

# mount -t glusterfs -oattribute-timeout=1,entry-timeout=1
web3:/datos /mnt/datos
```

3. Agregar el nuevo nodo (web3) en el Pool de la capa de balanceo, tanto pasivo como activo

```
backend servers
       balance roundrobin
       option httpchk HEAD/HTTP/1.0
       cookie SERVERID insert indirect nocache
       server s1 web1:80 check cookie s1
       server s1 web2:80 check cookie s2
       server s1 web3:80 check cookie s3
```

Al reiniciar el servicio de Haproxy en la capa de balanceo, el nuevo servidor comenzara a responder peticiones, una vez que los datos del sistema de Archivos compartido se encuentren actualizados. El proceso es el mismo por cada nodo Web extra que se agregue a la capa.

3.6 Plan de pruebas y resultados

Durante el proceso de prueba se comenzó por validar que cada uno de los componentes del cluster soportara fallas físicas, como también su capacidad de regresar a operación sin afectar al aplicativo Moodle, con estas pruebas es posible determinar el grado de HA del cluster.

Posteriormente se realizaron pruebas de balanceo de carga simulando gran número de peticiones al cluster y observando como afectaba el performance de los nodos, con los resultados obtenidos es posible determinar cuál es el algoritmo de balanceo ideal para la arquitectura propuesta.

3.6.1 Pruebas de alta disponibilidad

Se aplicaron intencionalmente varios escenarios de falla a los nodos de Balanceo, cada suceso se repitió en 10 ocasiones y se calculó el promedio de tiempo en el que el servicio se restablecía por completo, también se revisó la arquitectura completa en busca de fallas a nivel aplicación.

Simulando intencionalmente varios escenarios de falla a los nodos web, se validó el comportamiento a nivel aplicativo (persistencia de sesión) y se contabilizó el tiempo de failback de los nodos afectados, así como su capacidad de servir de nuevo peticiones. Se validó el tiempo de failback al GlusterFS y su afectación a la red.

A nivel base de datos se aplicaron intencionalmente situaciones de falla, validando la integridad de los datos, la ausencia de Split Brain, la velocidad de respuesta del aplicativo y la persistencia de sesión.

Finalmente se simuló el peor escenario posible de falla dentro del cluster, provocando fallas en los elementos que no cuentan con HA, se validó el funcionamiento del aplicativo así como el comportamiento de cada uno de los componentes de la arquitectura.

Pruebas de alta disponibilidad sobre la capa de balanceo

Falla física en el balanceador 1

Se aplicó un apagado inesperado, cortando el suministro de energía eléctrica en el nodo maestro, tras un heartbeat fallido de 5 segundos el nodo esclavo pasa a ser maestro y reanuda el servicio.

Número de Prueba	**Promedio Recuperación de servicio (seg.)**	**Persistencia de Sesión**
1	8.6	Si
2	8.8	Si
3	9	Si
4	8.1	Si
5	8.2	Si
6	8.8	Si
7	8.8	Si
8	8.5	Si
9	8.4	Si
10	8.5	Si
Promedio	**8.57**	

Tabla 11: Tiempo de reanudación de servicio en el nodo 2 de balanceo.

Tras 10 intentos se obtuvo un promedio de reanudación del servicio de 8.57 segundos. Durante éste periodo de tiempo el cluster no respondió a peticiones. A nivel usuario, el navegador solo presento un tiempo de carga mayor sin mostrar error en la pantalla y conservando la sesión de usuario.

Failback del Balanceador 1

Se restableció la corriente en el balanceador 1, y se contabilizo el tiempo total que le tomo al nodo1 su inicio de servicios y la adición nuevamente al cluster.

Numero de Prueba	Failback del Balanceador1 (seg.)
1	122.2
2	122.2
3	125.1
4	125.4
5	125.2
6	125.5
7	125.2
8	125.3
9	125.4
10	125.5
Promedio	**124.7**

Tabla 12: Tiempo de reanudación de servicio en el nodo 1 de balanceo.

Al balanceador 1 le llevó un promedio de 124 segundos regresar de su estado de falla y ser de nuevo parte del cluster, durante éste proceso la disponibilidad del servicio no se vio afectada en ningún momento, ya que el balanceador 2 es el que se encuentra sirviendo a peticiones.

Falla física en el balanceador 2

Ahora se simuló una falla de energía eléctrica en el balanceador 2, el cual en este momento es el nodo maestro y es el, quien responde a peticiones.

Numero de Prueba	Promedio Recuperación de servicio (seg.)	Persistencia de Sesión
1	8.9	Si
2	9.2	Si
3	9.3	Si
4	9.1	Si
5	9.4	Si
6	8.5	Si
7	8.7	Si
8	8.7	Si
9	8.9	Si
10	9.2	Si
Promedio	**8.99**	

Tabla 13: Tiempo de reanudación de servicio en el nodo 1 de balanceo.

El servicio se movió al nodo 1 y estuvo listo para servir a peticiones en un promedio de 8.99

segundos con persistencia de sesión, durante éste tiempo el cluster no fue capaz de recibir peticiones, a nivel usuario este proceso de failover se vio como un tiempo de espera mayor en el navegador.

Failback del balanceador 2

Al restablecer la energía eléctrica en el balanceador 2, se contabilizo el tiempo total que le tomó al nodo el proceso de booteo, inicio de servicios y la adición nuevamente al cluster.

Número de Prueba	Failback del Balanceador 2 (seg.)	Afectación
1	129.7	No
2	132.2	No
3	132.8	No
4	129.5	No
5	128.5	No
6	128.6	No
7	129.4	No
8	130.8	No
9	130.2	No
10	129.7	No
Promedio	**130.14**	

Tabla 14: Tiempo de reanudación de servicio en el nodo 2 de balanceo.

El balanceador 2 fue ingresado de nuevo al cluster en un promedio de 130 segundos, sin verse afectado el servicio, y quedando a la espera de algún nuevo evento en el cluster.

Pruebas de alta disponibilidad sobre la capa web

Se simuló una falla eléctrica 5 ocasiones en cada uno de los nodos, sin presentarse afectación alguna, de igual forma se produjo de manera deliberada un error en el servicio de apache, dado que la capa de balanceo realiza un check de servicio, se dejó de enviar peticiones a este nodo afectado, logrando con esto la continuidad del servicio.

.

Numero de Prueba	Tipo de Prueba	Tiempo de respuesta (seg.)	Afectación
1	Falla de Hardware	4.4	ninguna
2	Falla de Hardware	4.5	ninguna
3	Falla de Hardware	4.6	ninguna
4	Falla de Hardware	4.5	ninguna
5	Falla de Hardware	4.5	ninguna
6	Crash del proceso httpd	4.4	ninguna
7	Crash del proceso httpd	4.5	ninguna
8	Crash del proceso httpd	4.5	ninguna
9	Crash del proceso httpd	4.5	ninguna
10	Crash del proceso httpd	4.6	ninguna

Tabla 15: Afectación de servicio en capa web.

Prueba de alta disponibilidad en persistencia de sesión

Se simularon en 10 ocasiones un evento de falla en alguno de los nodos web que tenían una sesión activa en el aplicativo y en los que se estaba realizando alguna acción. Una vez hecho esto se validó que Moodle no perdiera la sesión que había generado.

Numero de Prueba	Acción	Persistencia de Sesión	Tiempo de Carga
1	Login a Página de inicio	si	4.8
2		si	4.4
3	Agregar comentario a entrada	si	4.6
4		si	4.6
5	Creación de nuevo Temario	si	4.7
6		si	4.9
7	Upload de foto	si	4.8
8		si	4.8
9	Realización de Examen	si	4.8
10		si	4.8

Tabla 16: Persistencia de sesión en capa web.

En todos los casos, sin importar la acción simulada, se conservó la sesión, con un tiempo de respuesta menor a 5 segundos en resolver la petición del Usuario.

Prueba de Failback en GlusterFS

Comprobación del tiempo necesario que le toma a cada uno de los nodos realizar un failback, después de que Moodle escribió datos.

Numero de Prueba	Acción	Tiempo de Sincronización	Promedio
1	Creación de Nuevo Curso	15.4	
2		16.5	15.95
3	Carga de 10 Usuarios Nuevos	28.6	
4		27.7	28.15
5	Upload de pdfs por 10 usuarios	60.5	
6		58.5	59.5
7	20 Entradas nuevas en Foro	55.4	
8		56.6	56
9	Creación de contenido multimedia	80.4	
10		89.7	85.05

Tabla 17: Resultados de tiempo de failback a GlusterFS.

Se observó un incremento del tiempo de Failback según el tamaño de los elementos nuevos que se agregaron durante el tiempo que este nodo permaneció fuera del cluster, un comportamiento lógico, ya que según la arquitectura planeada, todos los nodos web comparten el mismo switch de comunicación, por lo que el reingreso de un nodo puede causar lentitud en las respuestas hacia el cliente. Durante éste proceso el nodo con la acción de failback permaneció fuera del cluster hasta el momento que sus datos fueron sincronizados, por lo que en ninguno de los casos se presentó Split-brain, ni una afectación real en el servicio.

Pruebas de alta disponibilidad sobre la capa base de datos

Se simuló una falla eléctrica en ambos nodos, uno a la vez en 5 ocasiones, durante todas las pruebas, el servicio respondió de forma normal, ya que el nodo que se mantenía arriba no sufrió ningún evento de falla en el servicio de base de datos. Los tiempos de failback e integridad de datos se muestran en la *Tabla 18.*

Numero de Prueba	Acción	Tiempo de Failback	Promedio	Integridad de Datos
1	Creación de nuevo curso	5.7	5.45	SI
2		5.2		
3	Carga de 10 Usuarios Nuevos	7	6.9	SI
4		6.8		
5	Eliminación de 10 Usuarios	5.8	6.15	SI
6		6.5		
7	20 Entradas nuevas en Foro	6.4	6.25	SI
8		6.1		
9	Búsqueda de usuarios	5.9	5.8	SI
10		5.7		

Tabla 18. Resultados de tiempo de failback e integridad de datos en BD.

Cabe mencionar que en ninguno de los casos el nodo que realizo failback respondió a peticiones hasta tener sus datos sincronizados con el nodo activo, esto para evitar problemas de Split-brain. No se presentó corrupción de datos.

Pruebas a la arquitectura completa.

Crash en el servicio de HaProxy. Se simuló un crash en el servicio de HaProxy, en el nodo 1 de la capa de balanceo, deteniendo el servicio de balanceo en el nodo, pero manteniendo los beats entre nodos.

Resultados Obtenidos

En todas las ocasiones el nodo continuo arriba ya que las interfaces de red siguen mandando beats entre ellas y el demonio de HA no detecta falla alguna en el, dado que el servicio de balanceo fue detenido, todo recurso en el cluster se vuelve inaccesible y es necesario la intervención del administrador.

Crash del servicio de GlusterFS. Se simuló una falla en el servicio de GlusterFS, deteniendo el servicio.

Resultados Obtenidos:

Dado que el parámetro directory root del servidor web apache está apuntando la ruta montada con GlusterFS, el servicio web no inicio en ninguno de los intentos realizados, por lo que el balanceador no envía peticiones a ese nodo, el servicio no se ve afectado ya que los otros nodos se encuentran disponibles.

Corrupción de Datos en GaleraBD. Se simuló un evento de corrupción de datos, escribiendo datos aleatorios en el Sistema de Archivos donde está siendo consultada la base de datos.

Resultados Obtenidos:

El servicio deja de ser accesible solamente en este nodo y es necesaria la intervención del administrador para sincronizar nuevamente la información de la base de forma manual, debido a que se presenta un error de Split-Brain, dado que el balanceador realiza un check de salud en el servicio, sólo deja de enviar peticiones al nodo afectado, provocando que el servicio continué sin falla en el nodo extra.

Caída del Switch de la LAN. Se simuló una falla eléctrica en el switch que interconecta la LAN del Cluster.

Resultados Obtenidos:
Todos los elementos debajo de la capa de balanceo son inaccesibles, por lo que el servicio se ve afectado, es necesaria la intervención del administrador para revisar la falla

Falla en la Energía Eléctrica. Se simulo falla eléctrica en todos los nodos del cluster.

Resultados Obtenidos:

Ningún componente es accesible, no hay servicio. Es necesario contar con UPS y fuentes redundantes para que el cluster soporte una falla de este tipo.

Kernel Panic en Sistema Operativo. Se provocó intencionalmente en Kernel Panic en Linux.

Resultados Obtenidos:

El cluster soporta de forma correcta este tipo de falla en el sistema operativo, realojando los recursos en el nodo que se encuentra sano y conservando la integridad de datos.

Errores en la Aplicación Moodle. Se provocaron intencionalmente errores en el aplicativo.

Resultados Obtenidos:

El cluster no detecta este tipo de error, la aplicación continua sirviendo a peticiones pero con errores.

En el apartado de recomendaciones, se listan las posibles soluciones a los escenarios antes expuestos y en los que el cluster no fue capaz de continuar otorgando el servicio de forma correcta.

3.6.2 Pruebas de balanceo de carga

Se realizaron pruebas de tiempo de respuesta al aplicativo, en distintos escenarios de concurrencia de conexiones activas, utilizando la utilidad “ab” la cual viene incluida en la instalación por default del servidor web apache. También se validó el performance de cada uno de los nodos que integran el cluster. Todas estas pruebas se realizaron con cada uno de los algoritmos de balanceo configurados.

Simulación de concurrencia con el benchmark AB

Resultados Obtenidos:

3.6) 100 peticiones y 90 conexiones simultáneas a la web por petición usando 2 nodos webs

Tiempo total de la prueba LC: **14.349 segundos**

Tiempo total de la prueba RR: **14.670 segundos**

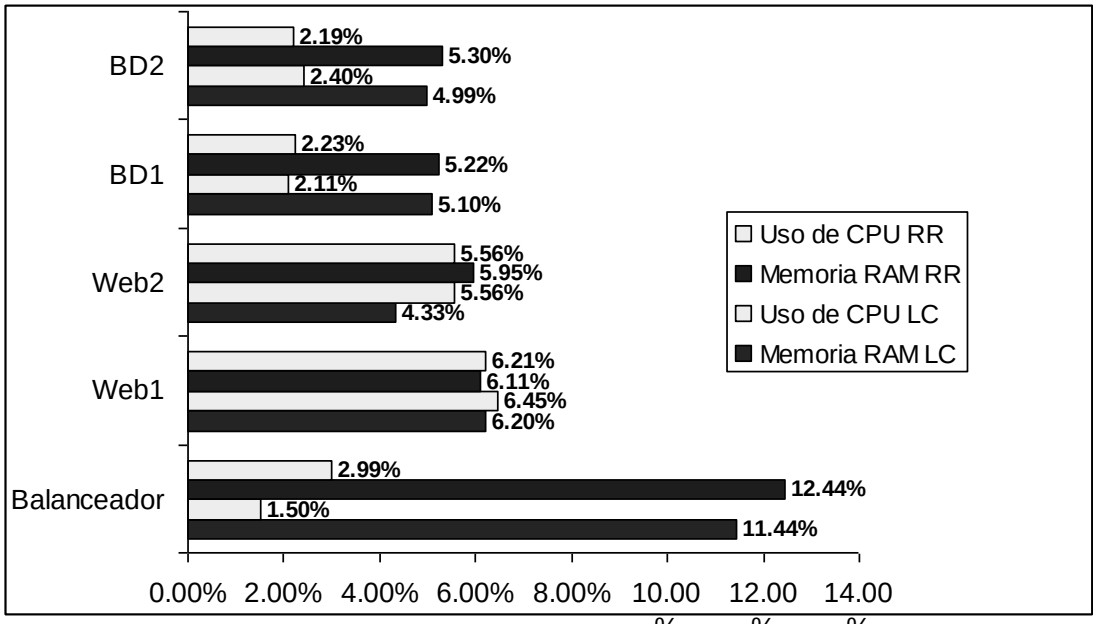

Tabla 19: Simulación de 100 peticiones y 90 conexiones simultáneas usando 2 nodos

1.2) 100 peticiones y 90 conexiones simultaneas a la web por petición usando 4 nodos webs

Tiempo total de la prueba LC: **9.465 segundos**

Tiempo total de la prueba RR: **9.653 segundos**

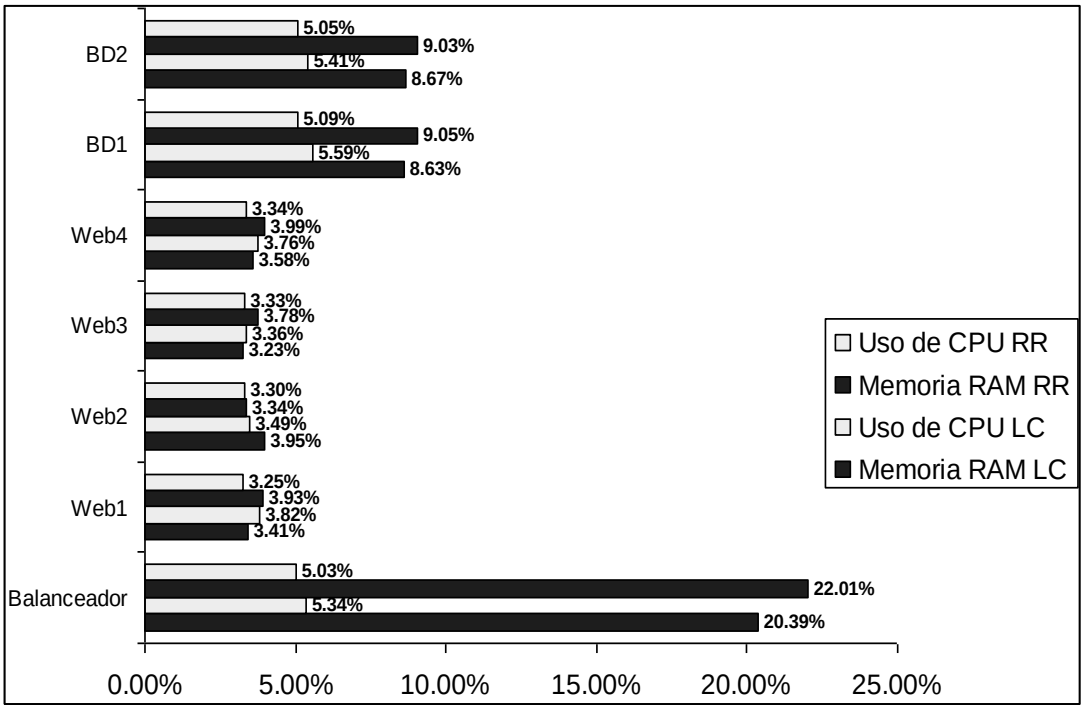

Tabla 20: Simulación de 100 peticiones y 90 conexiones simultáneas usando 4 nodos

2.1) 10000 peticiones y 5 conexiones simultaneas a la web por petición usando 2 nodos webs

Tiempo total de la prueba LC: **146.887 segundos**

Tiempo total de la prueba RR: **251.159 segundos**

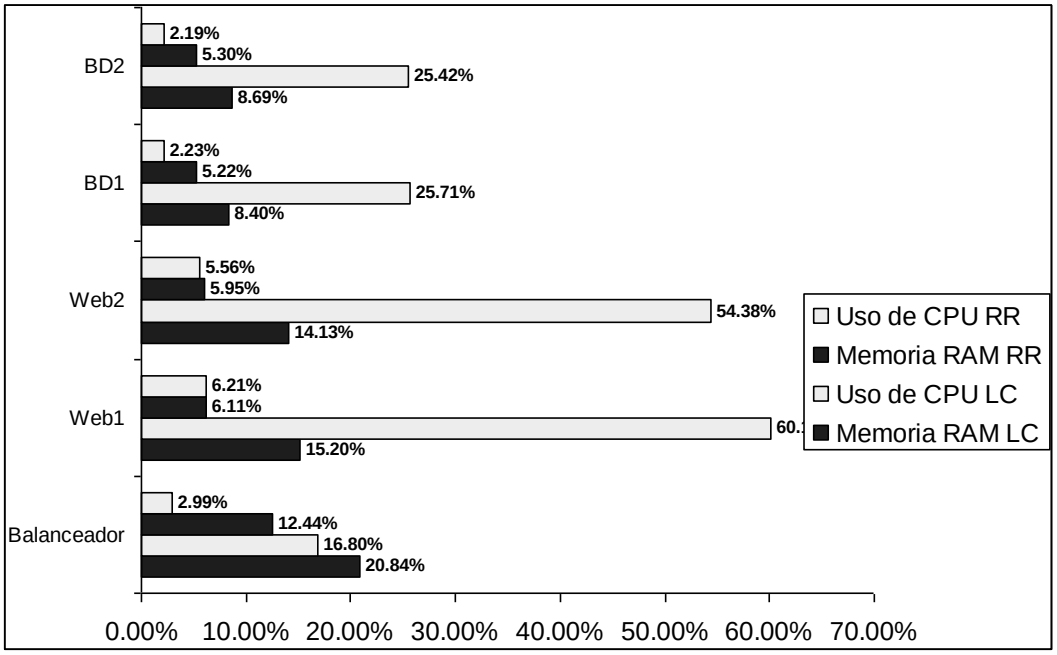

Tabla 21: Simulación de 10000 peticiones y 5 conexiones simultáneas usando 2 nodos

2.2) 10000 peticiones y 5 conexiones simultaneas a la web por petición usando 4 nodos web

Tiempo total de la prueba LC: **113.940 segundos**

Tiempo total de la prueba RR: **114.440 segundos**

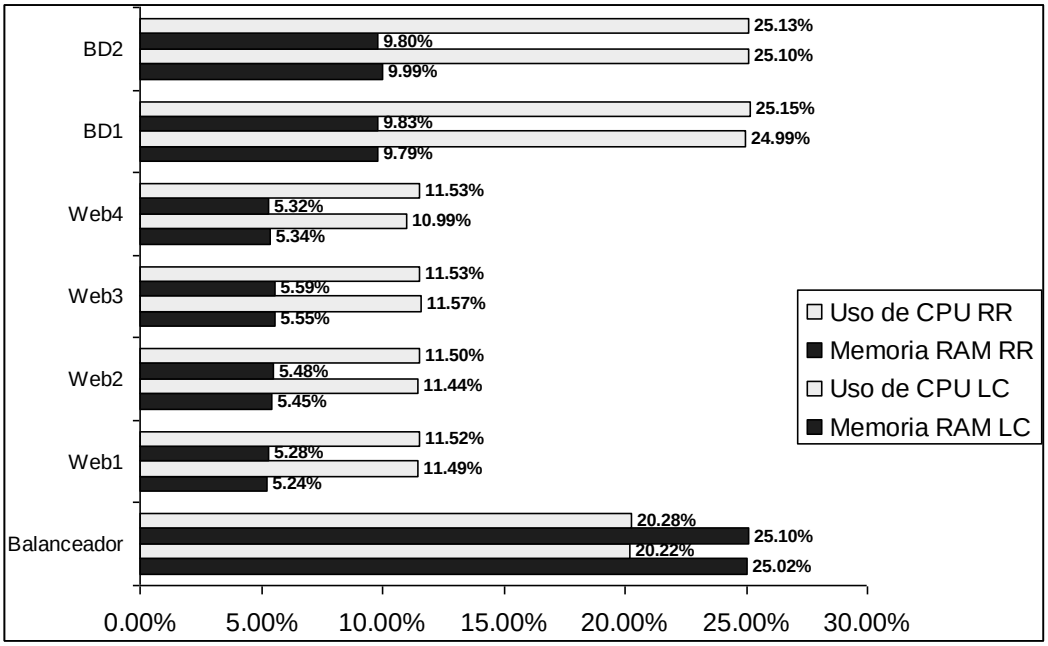

Tabla 22: Simulación de 10000 peticiones y 5 conexiones simultáneas usando 4 nodos

3.1) 1000 peticiones y 500 conexiones simultaneas a la web por petición usando 2 nodos web

Tiempo total de la prueba LC: **145.412 segundos**

Tiempo total de la prueba RR: **124.738 segundos**

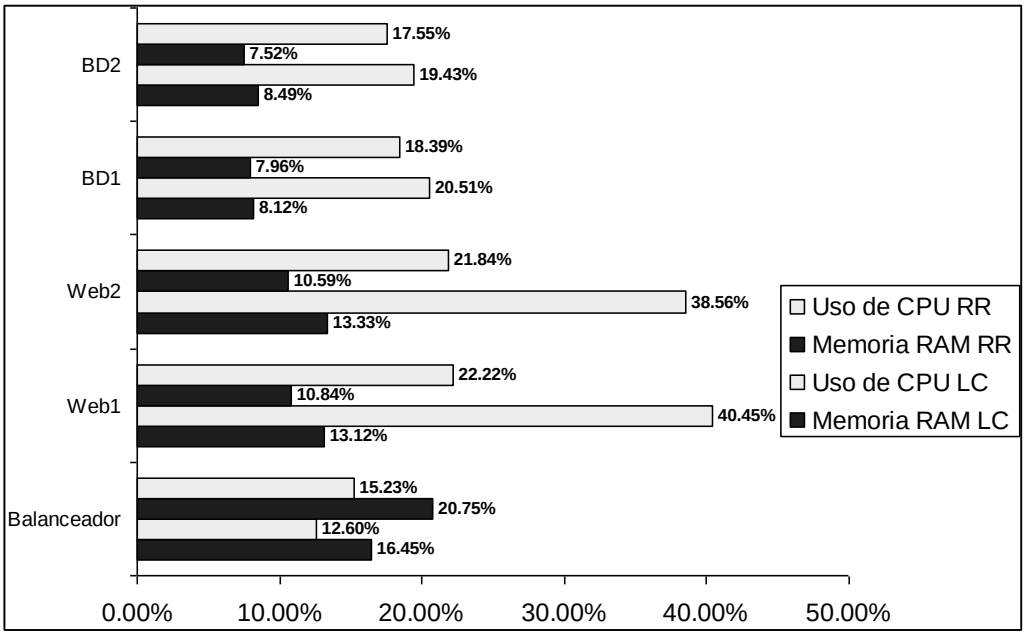

Tabla 23: Simulación de 1000 peticiones y 500 conexiones simultáneas usando 2 nodos

3.2) 1000 peticiones y 500 conexiones simultaneas a la web por petición usando 4 nodos web

Tiempo total de la prueba LC: **93.246 segundos**

Tiempo total de la prueba RR: **96.050 segundos**

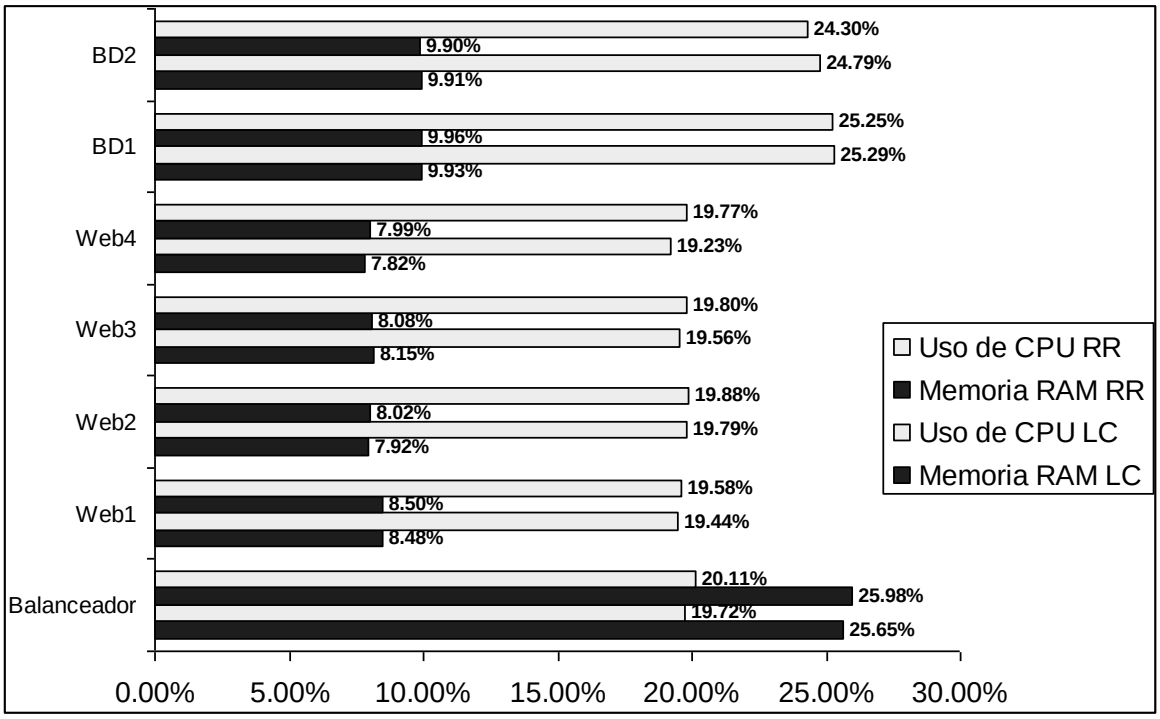

Tabla 24: Simulación de 1000 peticiones y 500 conexiones simultáneas usando 4 nodos

4.1) 10000 peticiones y 100 conexiones simultaneas a la Web por petición usando 2 Nodos Web

Tiempo total de la prueba LC: **1463.543**

Tiempo total de la prueba RR: **1461.478**

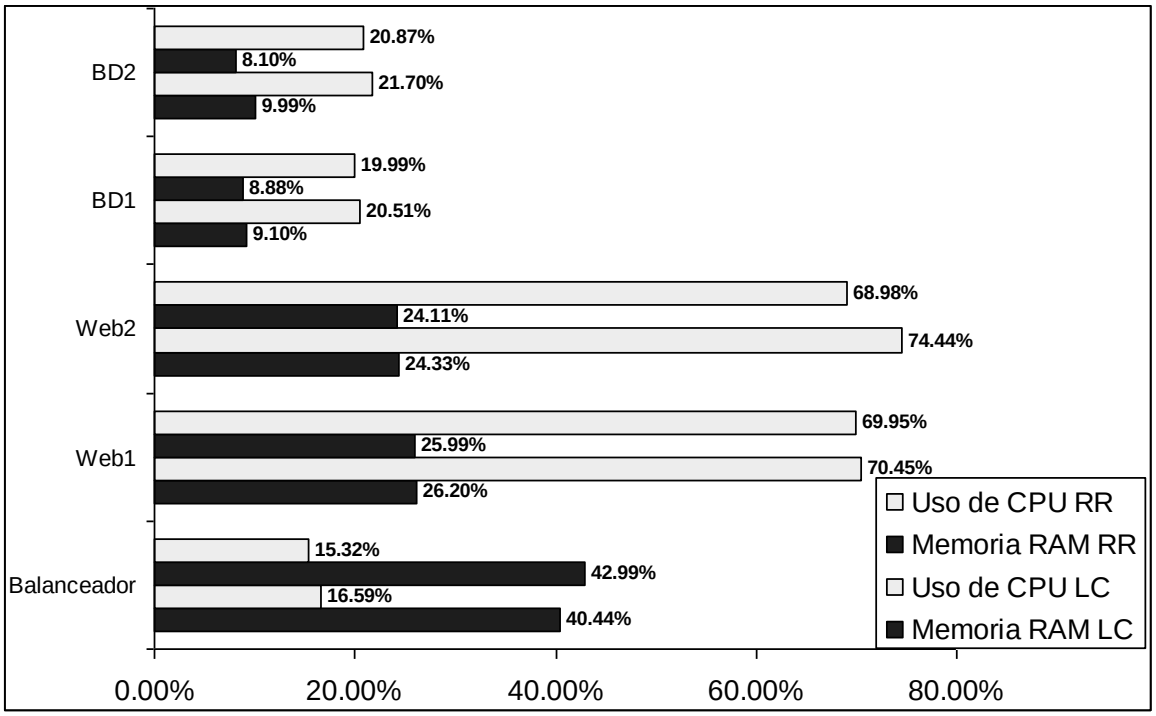

Tabla 25: Simulación de 10000 peticiones y 100 conexiones simultáneas usando 2 nodos

4.2) 10000 peticiones y 100 conexiones simultaneas a la Web por petición usando 4 Nodos Web

Tiempo total de la prueba LC**: 933.095**

Tiempo total de la prueba RR**: 974.499**

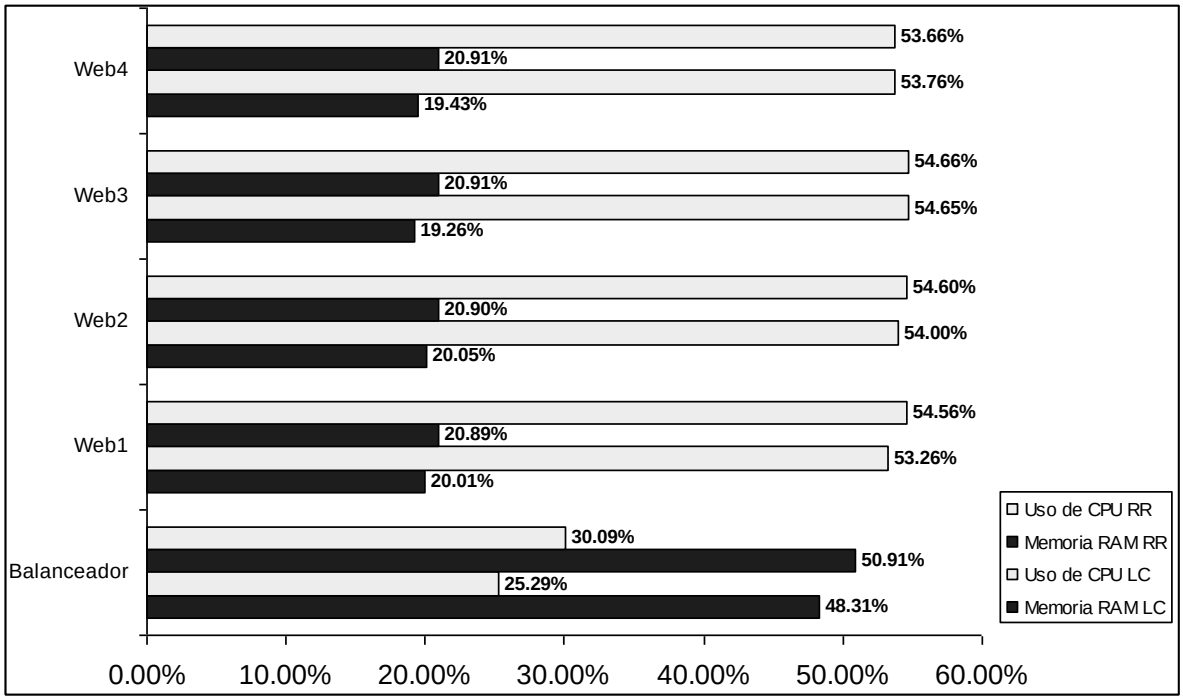

Tabla 26: Simulación de 10000 peticiones y 100 conexiones simultáneas usando 4 nodos

Resultados

En todas las pruebas anteriores se puede observar que el balanceo por LC termino en menos tiempo de resolver las peticiones, y usando un performance inferior en todos los nodos que el balanceo por RR.

El porcentaje de uso de recursos en los nodos aumentaba como se esperaba conforme las peticiones se incrementaban, sin importar el algoritmo de balanceo usado.

El nodo activo de la capa de balanceo y los nodos web fueron los que durante todas las pruebas presentaron un mayor uso de recursos.

Agregando 2 nodos más a la capa web se produce una reducción en el tiempo de respuesta a peticiones, sin embargo aumenta el uso de recursos en el equipo de balanceo y en los nodos de base de datos, esto debido a que los nodos deben de resolver peticiones de los nodos extras.

Es importante mencionar que después de cada una de las pruebas se cerraron todas las conexiones establecidas y se reiniciaron los servidores de la capa Web, esto para obtener resultados más precisos.

3.7 Simulación de ataque de denegación de servicio (DoS)

Para realizar esta prueba se utilizó la herramienta LOIC, la cual se encuentra disponible para su descarga en http://sourceforge.net/projects/loic/ éste software es capaz de enviar una gran cantidad de paquetes TCP, paquetes UDP o peticiones HTTP con la finalidad de determinar cuál es la cantidad máxima de peticiones por segundo que puede resolver un objetivo antes de dejar de funcionar debido a la carga excesiva.

Ha sido utilizado en varias ocasiones para realizar ataques reales a diferentes organizaciones, y causando afectación a servidores productivos.

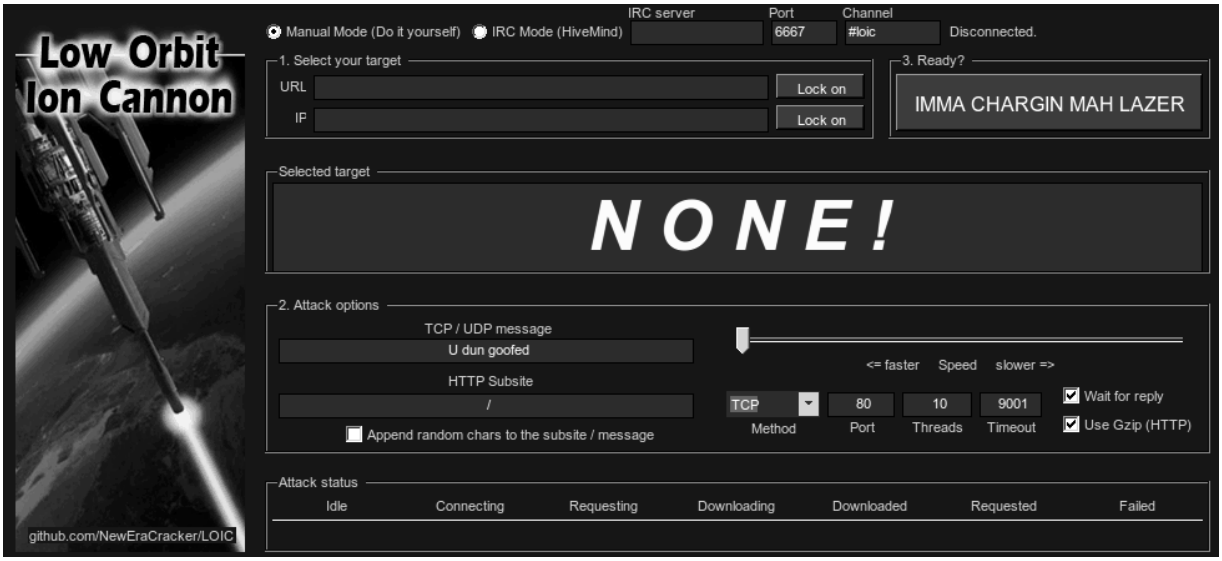

Figura 43: Software LOIC

a) Se realizaron las pruebas de estrés con LOIC durante 30 minutos usando los parámetros por default , seleccionando el método HTTP, y obteniendo durante el ataque los siguientes resultados:

Arquitectura con 2 nodos en capa web

Nodos	Memoria RAM	Uso de CPU
Balanceador	88.88%	87.34%
Web1	96.64%	96.81%
Web2	96.83%	96.82%
BD1	90.10%	90.19%
BD2	90.41%	90.35 %

Tabla 27: Simulación de ataque DoS usando 2 nodos

Se validó el tiempo acceso a aplicativo Moodle cada 15 minutos, ejecutando distintas acciones en el mismo y obteniendo los siguientes resultados:

Tiempo transcurrido de Ataque	Acción	Persistencia de Sesión	Tiempo de Carga (segundos)
15 min	Login a Página de inicio	si	9.2
30 min		si	10.2
15 min	Agregar comentario a entrada	si	9.8
30 min		si	10.1
15 min	Creación de nuevo Temario	si	9.3
30 min		si	11.1
15 min	Upload de foto	si	9.6
30 min		si	10.4
15 min	Realización de Examen	si	9.4
30 min		si	10.2

Tabla 28: Tiempo de respuesta del aplicativo durante ataque

El cluster funciono de forma correcta durante el ataque simulado, sin embargo aumentó de manera considerable el tiempo de carga del aplicativo. El balanceador realizó de forma correcta la distribución de las peticiones en los nodos web y en los nodos de base de datos.

Se procedió a Agregar 2 nodos más a la capa Web:

Nodos	Memoria RAM	Uso de CPU
Balanceador	90.55%	93.44%
Web1	90.64%	94.81%
Web2	90.83%	94.82%
Web3	90.95%	94.97%
Web4	90.21%	93.89%
BD1	92.40%	90.69%
BD2	92.59%	90.75 %

Tabla 29: Simulación de ataque DoS usando 4 nodos

La tabla anterior muestra una disminución en el uso de RAM y CPU en la capa web, ésto debido a los dos nodos extras que fueron agregados.

Nuevamente se validó el tiempo acceso a aplicativo Moodle cada 15 minutos, ejecutando distintas acciones en el mismo y obteniendo los siguientes resultados:

Tiempo transcurrido de Ataque	Acción	Persistencia de Sesión	Tiempo de Carga (segundos)
15 min	Login a Página de inicio	si	7.4
30 min		si	6.5
15 min	Agregar comentario a entrada	si	5.8
30 min		si	6.2
15 min	Creacion de nuevo Temario	si	6.5
30 min		si	6.4
15 min	Upload de foto	si	6.6
30 min		si	6.4
15 min	Realización de Examen	si	6.4
30 min		si	6.2

Tabla 30: Tiempo de respuesta del aplicativo durante ataque

El cluster funcionó de forma correcta durante el ataque simulado, aumentando solo un poco los tiempos de respuesta de la aplicación y disminuyendo el tiempo de carga con respecto a la prueba con solo 2 nodos, el balanceador distribuía de forma correcta la carga entre los 4 nodos web y los 2 nodos de base de datos durante el ataque.

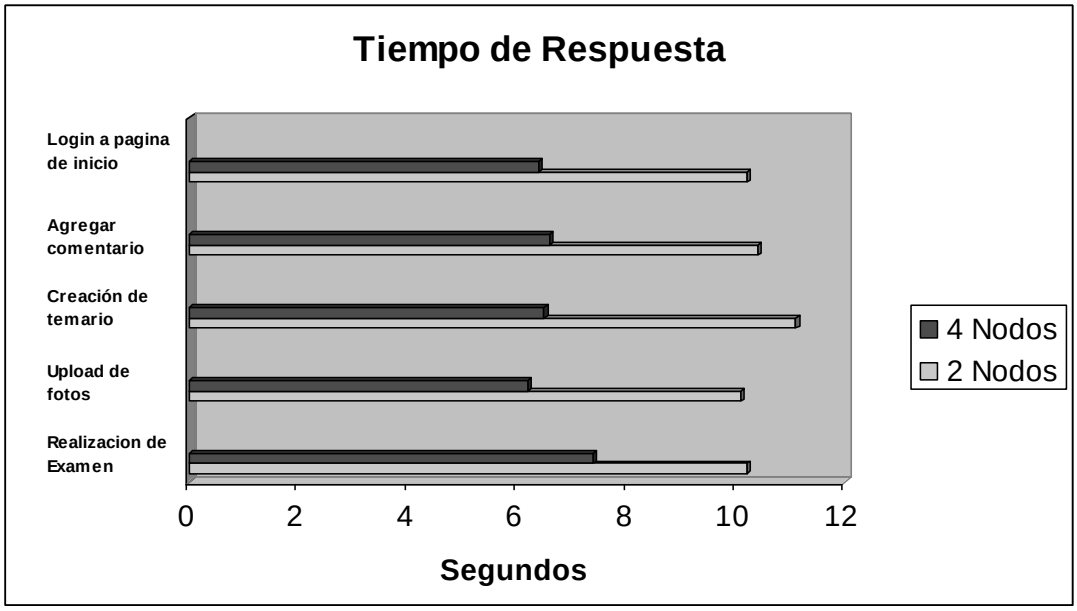

Tabla 31: Tiempos de respuesta durante ataque DoS (parámetros por default)

b) Se realizaron las pruebas de estrés con LOIC durante 30 minutos, aumentando el número de treads a 100, seleccionando el método HTTP, y obteniendo durante el ataque los siguientes resultados:

Arquitectura con 2 nodos en Capa Web

Nodos	Memoria RAM	Uso de CPU
Balanceador	100%	100%
Web1	99.43%	99.91%
Web2	99.36%	99.41%
BD1	99.20%	99.74%
BD2	99.38%	99.65 %

Tabla 32: Simulación de ataque DoS usando 4 nodos

Se validó el tiempo acceso a aplicativo Moodle cada 15 minutos, obteniendo los siguientes resultados:

Tiempo transcurrido de Ataque	Acción	Persistencia de Sesión	Tiempo de Carga (segundos)
15 min	Login a Pagina de inicio	si	18.8
30 min		---	--
15 min	Agregar comentario a entrada	si	18.4
30 min		---	--
15 min	Creacion de nuevo Temario	si	18.7
30 min		---	--
15 min	Upload de foto	si	18.3
30 min		---	--
15 min	Realización de Examen	si	18.2
30 min		---	--

Tabla 33: Tiempo de respuesta del aplicativo durante ataque

El balanceador dejó de responder a los 12 minutos de haber iniciado el ataque, debido al aumento excesivo de recursos, fue necesario aplicar un reinicio al nodo. La prueba se fue realizada una segunda ocasión con el mismo resultado.

Agregando 2 nodos más a la capa web

Nodos	Memoria RAM	Uso de CPU
Balanceador	100.00%	100.00%
Web1	96.54%	99.44%
Web2	96.65%	99.26%
Web3	96.47%	99.78%
Web4	96.39%	99.55%
BD1	97.60%	95.28%
BD2	97.78%	96.19 %

Tabla 34: Simulación de ataque DoS usando 4 nodos

Se validó el tiempo acceso a aplicativo Moodle cada 15 minutos, ejecutando las siguientes acciones:

Tiempo transcurrido de Ataque	Acción	Persistencia de Sesión	Tiempo de Carga (segundos)
15 min	Login a Página de inicio	si	25.5
30 min		no	23.3
15 min	Agregar comentario a entrada	si	25.7
30 min		si	22.2
15 min	Creación de nuevo Temario	si	32.8
30 min		no	33.9
15 min	Upload de foto	si	43.4
30 min		si	46.2
15 min	Realización de Examen	si	36.9
30 min		no	35.2

Tabla 35: Tiempo de respuesta del aplicativo durante ataque

En esta ocasión el balanceador y las demás capas del cluster soportaron los 30 minutos en que fue simulado el ataque, sin embrago se notó un aumento considerable en la carga de Moodle y continuas desconexiones a la base de datos y servidores webs.

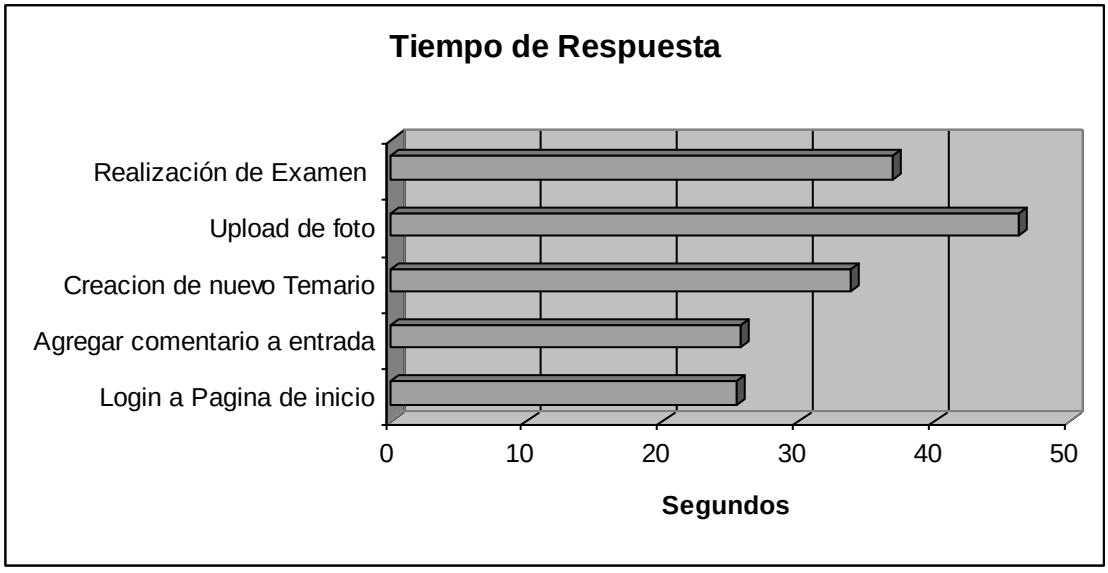

Tabla 36: Tiempos de respuesta durante ataque DoS (100 treads)

Recomendaciones

A continuación se listan las recomendaciones que se sugieren para una alta disponibilidad nivel 5 dentro de una arquitectura de cluster, tomando en cuenta las limitantes con las que cuenta el proyecto.

1.- Durante las pruebas se simularon fallas irrecuperables en el servicio de Haproxy, para solucionar ésta limitante en el cluster es necesario utilizar un script de monitoreo de procesos, el cual se encarga de revisar si un proceso se encuentra operando de forma correcta, y si no, se encargará de reiniciarlo.

2.- Es posible que los equipos activos de red fallen, para lograr alta disponibilidad dentro de una arquitectura crítica, es necesario contar con switches y adaptadores de red redundantes, éstos últimos dentro del sistema operativo deben de configurarse como un bonding en modo failover, logrando que, en caso de presentar una falla por parte del switch o algún adaptador, la red continué operando en el servidor.

El uso de UPSs o contar con energía eléctrica de respaldo es vital en todo sistema crítico, es necesario que el cluster cuente con éste tipo de protección, además de un plan de acción en caso de que el área donde se encuentran los servidores presente algún daño físico grave y que todos los componentes del cluster queden inutilizables. En éste caso se propone contar con un sitio de respaldo en CAMPO 1, el cual será una copia básica de la arquitectura productiva y que se encontrará en modo pasivo, este recibirá copias incrementales de la capa WEB y capa de base de datos diariamente en un horario nocturno, con el fin de mantener actualizado el sitio remoto y preparado en caso de que sea necesario su uso.

Es conveniente contar con algún monitoreo que revise cada determinando tiempo la funcionalidad del aplicativo (Moodle), validando el acceso a los recursos específicos de la aplicación, y en caso de presentar fallas envíe alertas al responsable de ésta. Cabe mencionar que es necesario que la FESC cuente con un procedimiento completo para realizar cambios o actualizaciones dentro de la aplicación cuando esto sea necesario y en caso de presentarse

errores con nuevas versiones, realizar el reverso de la aplicación. Por último, es necesario que la FESC implemente en conjunto con el cluster una protección extra contra ataques DoS, debido a que el cluster por sí solo no ofrece una protección completa y la infraestructura continúa siendo vulnerable a este tipo de amenazas.

Conclusiones

En este trabajo fue presentada la implementación bajo un ambiente de pruebas de un cluster de Alta Disponibilidad y Balanceo de Carga para la protección de datos dentro de la FES CUAUTITLAN bajo un ambiente de prueba, se demostró que es posible lograr alta disponibilidad a nivel físico y nivel aplicación, usando PCs normales y utilizando software libre para este fin. Lo que representa un ahorro considerable a comparación de soluciones propietarias.

A través de la elaboración del proyecto se comprobó que las soluciones de software libre que existen actualmente para proporcionar Alta Disponibilidad y Balanceo de Carga presentan una estabilidad y facilidad de configuración bastante alta, además de estar soportados y mantenidos por la comunidad OpenSource.

Se sometió a la arquitectura propuesta a varios escenarios de falla comunes, y se observó una alta tolerancia a fallos e integridad de los datos en todas las pruebas realizadas, con la totalidad de la arquitectura implementada se logró alcanzar un Nivel 4 de disponibilidad.

Durante las pruebas realizadas al cluster, fue posible comprobar que la arquitectura propuesta soporta una carga mayor de usuarios que la que pudiera llegar a soportar un solo servidor, esto da como resultado un tiempo de respuesta menor al consultar el aplicativo por parte del usuario.

Después de varias pruebas de carga con distintas configuraciones, fue posible validar que el algoritmo de balanceo óptimo para Moodle bajo esta arquitectura propuesta es el de Least Conection, dando los mejores tiempos de respuesta.

Se comprobó que la arquitectura de cluster propuesta por sí sola no es suficientemente robusta para soportar ataques de Denegación de Servicio, sin embrago, en las pruebas se pudo observar que el cluster seguía dando servicio durante el ataque de forma correcta durante el mayor tiempo posible, esto es vital ya que, da el tiempo necesario al administrador de tomar

las medidas pertinentes para contener el ataque.

Aplicando la arquitectura propuesta se redujo la posibilidad de pérdida o corrupción de datos, aun durante un ataque de denegación de servicio y aumentó la calidad del servicio al usuario final por lo que la hipótesis alternativa en la cual se busca lograr una diferencia significativa en la eliminación de ataques, fallas y contingencias fue valida.

Con los resultados obtenidos en esta tesis se espera que la FES CUAUTITLAN considere la futura implementación de ambientes en alta disponibilidad en sus sistemas críticos, y que este proyecto sirva como base para la elección de una correcta arquitectura.

Fuentes de información

- Barroso L.A, Dean J. and Hölzle U..,(2003), *Web Search for a Planet: The Google Cluster Architecture*, IEEE Computer Society, recuperado 2 de Enero 2013 de books.ithunder.org/google_clusters.pdf

- Dekker Arjan , Hobo Remco.,(Julio 4, 2005), *High Availability Services,* recuperado el 2 de Enero 2013 de http://staff.science.uva.nl/~delaat/rp/2004-2005/p29/report.pdf

- Gaston C. Hillar., (2004), *Servidores de Redes Diseño, Actualización y Reparación*, Argentina, ed. Hispano Americana S.A.

- Kopparapu Chandra., (2002), *L*oad balancing servers firewalls and caches, Estados Unidos, ed. Wily.

- Martínez M. and Bergés G.,(Marzo 2003), *Arquitectura de Clustering de Alta Disponibilidad,* recuperado el 10 de Enero 2013 de http://es.scribd.com/doc/2925766/acadelvsmemoria

- Sánchez López Tomas., (Febrero 2004), *Linux Server Clustering*, Universidad Politécnica de Valencia, recuperado el 3 de Enero 2013 de http://www.srcf.ucam.org/~tsl26/Analysis_on_Linux_Server_Clustering.pdf

- Stallings William., (2003), *Comunicaciones y Redes de Computadoras*, España, ed. Prentice Hall.

- Tanenbaum A. S., (1997) *Redes de Computadoras, México,* ed. Prentice Hall Hispanoamericana S.A.

- Tarreau Willy., (Septiembre 2006), *Making applications scalable with load balancing,* recuperado el 3 de Enero 2013 de http://www.exceliance.fr/sites/default/files/biblio/art-2006 making_applications_scalable_with_lb.pdf

- Wensong Zhang., *Linux Virtual Server for Scalable Network Services*, National Laboratory for Parallel & Distributed Processing Changsha, Hunan 41073, China, recuperado el 10 de Enero 2013 de http://www.linuxvirtualserver.org/ols/lvs.pdf

ANEXOS

I. Configuración de nodos de balanceo

```
global
        log 127.0.0.1   local0
        log 127.0.0.1   local1 notice
        maxconn 1024
        user haproxy
        group haproxy
        daemon

defaults
        log     global
        mode    http
        option  tcplog
        option  dontlognull
        retries 3
        option  redispatch
        maxconn 1024
        timeout connect 5000ms
        timeout client 50000ms
        timeout server 50000ms

listen mariadb_cluster_writes 0.0.0.0:13305
## A failover pool for writes to ensure writes only hit one node at a time.
        mode tcp
        option httpchk
        server galera-node01 192.168.3.20:3306 check port 9200
        server galera-node02 192.168.3.21:3306 check port 9200 backup

listen mariadb_cluster_reads 0.0.0.0:13304
## A load-balanced pool for reads to utilize all nodes for reads.
        mode tcp
        balance leastconn
        option httpchk
       server galera-node01 192.168.3.20:3306 check port 9200
        server galera-node02 192.168.3.21:3306 check port 9200

listen stats 0.0.0.0:1936
mode http
log global

maxconn 10

#clitimeout 100s
#srvtimeout 100s
#contimeout 100s
#timeout queue 100s

stats enable
stats hide-version
stats refresh 30s
stats show-node
stats auth admin:pug
stats uri /stats
```

```
frontend http-in
       bind *:80
       default_backend servers

#stylesheets

backend servers
       balance roundrobin
       option httpchk HEAD/HTTP/1.0
       cookie SERVERID insert indirect nocache
       server s1 web1:80 check cookie s1
       server s2 web2:80 check cookie s2
#      server s3 web3:80 check cookie s3
#      server s4 web4:80 check cookie s4

#backend servers
       #balance leastconn
       #option httpchk HEAD/HTTP/1.0
       #cookie SERVERID insert indirect nocache
       #server s1 web1:80 check cookie s1
       #server s1 web2:80 check cookie s2
       #server s3 web1:80 check cookie s3
       #server s4 web2:80 check cookie s4




listen statis 0.0.0.0:9000
## HAProxy stats web gui.
       mode http
       stats enable
       stats uri /haproxy_stats
       stats realm HAProxy\ Statistics
       stats auth haproxy:haproxy
       stats admin if TRUE
```

II. Script de revisión de Mariabd

```
#!/bin/bash
#
# Script to make a proxy (ie HAProxy) capable of monitoring Percona XtraDB Cluster
nodes properly
#
# Author: Olaf van Zandwijk <olaf.vanzandwijk@nedap.com>
# Author: Raghavendra Prabhu <raghavendra.prabhu@percona.com>
#
# Documentation and download: https://github.com/olafz/percona-clustercheck
#
# Based on the original script from Unai Rodriguez
#

if [[ $1 == '-h' || $1 == '--help' ]];then
    echo "Usage: $0 <user> <pass> <available_when_donor=0|1> <log_file>
<available_when_readonly=0|1> <defaults_extra_file>"
    exit
fi

# if the disabled file is present, return 503. This allows
# admins to manually remove a node from a cluster easily.
if [ -e "/var/tmp/clustercheck.disabled" ]; then
    # Shell return-code is 1
    echo -en "HTTP/1.1 503 Service Unavailable\r\n"
    echo -en "Content-Type: text/plain\r\n"
    echo -en "Connection: close\r\n"
    echo -en "Content-Length: 51\r\n"
    echo -en "\r\n"
    echo -en "Percona XtraDB Cluster Node is manually disabled.\r\n"
    sleep 0.1
    exit 1
fi

MYSQL_USERNAME="${1-usuario}"
MYSQL_PASSWORD="${2-pass}"
AVAILABLE_WHEN_DONOR=${3:-0}
ERR_FILE="${4:-/dev/null}"
AVAILABLE_WHEN_READONLY=${5:-1}
DEFAULTS_EXTRA_FILE=${6:-/etc/my.cnf}

#Timeout exists for instances where mysqld may be hung
TIMEOUT=10

EXTRA_ARGS=""
if [[ -n "$MYSQL_USERNAME" ]]; then
    EXTRA_ARGS="$EXTRA_ARGS --user=${MYSQL_USERNAME}"
fi
if [[ -n "$MYSQL_PASSWORD" ]]; then
    EXTRA_ARGS="$EXTRA_ARGS --password=${MYSQL_PASSWORD}"
fi
```

```
if [[ -r $DEFAULTS_EXTRA_FILE ]];then
    MYSQL_CMDLINE="mysql --defaults-extra-file=$DEFAULTS_EXTRA_FILE -nNE --
connect-timeout=$TIMEOUT \
                    ${EXTRA_ARGS}"
else
    MYSQL_CMDLINE="mysql -nNE --connect-timeout=$TIMEOUT ${EXTRA_ARGS}"
fi
#
# Perform the query to check the wsrep_local_state
#
WSREP_STATUS=$($MYSQL_CMDLINE -e "SHOW STATUS LIKE 'wsrep_local_state';" \
    2>${ERR_FILE} | tail -1 2>>${ERR_FILE})

if [[ "${WSREP_STATUS}" == "4" ]] || [[ "${WSREP_STATUS}" == "2" &&
${AVAILABLE_WHEN_DONOR} == 1 ]]
then
    # Check only when set to 0 to avoid latency in response.
    if [[ $AVAILABLE_WHEN_READONLY -eq 0 ]];then
        READ_ONLY=$($MYSQL_CMDLINE -e "SHOW GLOBAL VARIABLES LIKE 'read_only';" \
                    2>${ERR_FILE} | tail -1 2>>${ERR_FILE})

        if [[ "${READ_ONLY}" == "ON" ]];then
            # Percona XtraDB Cluster node local state is 'Synced', but it is in
            # read-only mode. The variable AVAILABLE_WHEN_READONLY is set to 0.
            # => return HTTP 503
            # Shell return-code is 1
            echo -en "HTTP/1.1 503 Service Unavailable\r\n"
            echo -en "Content-Type: text/plain\r\n"
            echo -en "Connection: close\r\n"
            echo -en "Content-Length: 43\r\n"
            echo -en "\r\n"
            echo -en "Percona XtraDB Cluster Node is read-only.\r\n"
            sleep 0.1
            exit 1
        fi
    fi
    # Percona XtraDB Cluster node local state is 'Synced' => return HTTP 200
    # Shell return-code is 0
    echo -en "HTTP/1.1 200 OK\r\n"
    echo -en "Content-Type: text/plain\r\n"
    echo -en "Connection: close\r\n"
    echo -en "Content-Length: 40\r\n"
    echo -en "\r\n"
    echo -en "Percona XtraDB Cluster Node is synced.\r\n"
    sleep 0.1
    exit 0
else
    # Percona XtraDB Cluster node local state is not 'Synced' => return HTTP 503
    # Shell return-code is 1
    echo -en "HTTP/1.1 503 Service Unavailable\r\n"
    echo -en "Content-Type: text/plain\r\n"
    echo -en "Connection: close\r\n"
    echo -en "Content-Length: 44\r\n"
    echo -en "\r\n"
```

```
    echo -en "Percona XtraDB Cluster Node is not synced.\r\n"
    sleep 0.1
    exit 1
fi
```

III. Servicio de revisión para MariaBD

```
# default: on
# description: mysqlchk
service mysqlchk
{
        disable = no
        flags = REUSE
        socket_type = stream
        port = 9200
        wait = no
        user = nobody
        server = /usr/bin/clustercheck
        log_on_failure += USERID
        only_from = 0.0.0.0/0
        per_source = UNLIMITED
}
```

```
#
# These groups are read by MariaDB server.
# Use it for options that only the server (but not clients) should see
#
# See the examples of server my.cnf files in /usr/share/mysql/
#

# this is read by the standalone daemon and embedded servers
[server]

# this is only for the mysqld standalone daemon
[mysqld]

Nodo1

#
# * Galera-related settings
#
[galera]
# Mandatory settings
#wsrep_provider=
#wsrep_cluster_address=
#binlog_format=row
#default_storage_engine=InnoDB
#innodb_autoinc_lock_mode=2
bind-address=0.0.0.0
#
# Optional setting
#wsrep_slave_threads=1
#innodb_flush_log_at_trx_commit=0

# this is only for embedded server
[embedded]

# This group is only read by MariaDB-5.5 servers.
# If you use the same .cnf file for MariaDB of different versions,
# use this group for options that older servers don't understand
[mysqld-5.5]

# These two groups are only read by MariaDB servers, not by MySQL.
# If you use the same .cnf file for MySQL and MariaDB,
# you can put MariaDB-only options here
[mariadb]
query_cache_size=0
binlog_format=ROW
default_storage_engine=innodb
innodb_autoinc_lock_mode=2
wsrep_provider=/usr/lib64/galera/libgalera_smm.so
wsrep_cluster_address=gcomm://192.168.3.21
wsrep_cluster_name='clusterbd'
wsrep_node_address='192.168.3.20'
```

```
wsrep_node_name='base2'
wsrep_sst_method=rsync
wsrep_sst_auth=root:tw1nn0r1tt@

[mariadb-5.5]
```

NODO 2

```
#
# * Galera-related settings
#
[galera]
# Mandatory settings
#wsrep_provider=
#wsrep_cluster_address=
#binlog_format=row
#default_storage_engine=InnoDB
#innodb_autoinc_lock_mode=2
bind-address=0.0.0.0
#
# Optional setting
#wsrep_slave_threads=1
#innodb_flush_log_at_trx_commit=0

# this is only for embedded server
[embedded]

# This group is only read by MariaDB-5.5 servers.
# If you use the same .cnf file for MariaDB of different versions,
# use this group for options that older servers don't understand
[mysqld-5.5]

# These two groups are only read by MariaDB servers, not by MySQL.
# If you use the same .cnf file for MySQL and MariaDB,
# you can put MariaDB-only options here
[mariadb]
query_cache_size=0
binlog_format=ROW
default_storage_engine=innodb
innodb_autoinc_lock_mode=2
wsrep_provider=/usr/lib64/galera/libgalera_smm.so
wsrep_cluster_address=gcomm://192.168.3.20
wsrep_cluster_name='clusterbd'
wsrep_node_address='192.168.3.21'
wsrep_node_name='base2'
wsrep_sst_method=rsync
wsrep_sst_auth=root:tw1nn0r1tt@

[mariadb-5.5]
```

V. Configuración Moodle

```
<?php  // Moodle configuration file

unset($CFG);
global $CFG;
$CFG = new stdClass();

$CFG->dbtype    = 'mariadb';
$CFG->dblibrary = 'native';
$CFG->dbhost    = '192.168.3.15';
$CFG->dbname    = 'moodle';
$CFG->dbuser    = 'root';
$CFG->dbpass    = 'tw1nn0r1tt@';
$CFG->prefix    = 'mdl_';
$CFG->dboptions = array (
  'dbpersist' => 0,
  'dbport' => 13305,
  'dbsocket' => '',
);

$CFG->wwwroot   = 'http://192.168.1.150/moodle';
$CFG->dataroot  = '/var/datos/moodledata';
$CFG->admin     = 'admin';

$CFG->directorypermissions = 0777;

require_once(dirname(__FILE__) . '/lib/setup.php');

// There is no php closing tag in this file,
// it is intentional because it prevents trailing whitespace problems!
```

VI. Script instala balanceo

```
#!/bin/bash
echo --- Configurando /etc/hosts ----
read -p "hostname Nodo1: " h1
read -p "IP Nodo1: " n1
read -p "hostname Nodo1: " h2
read -p "IP Nodo1: " n2
echo $h1        $n1 >>/etc/hosts
echo $h2        $n2 >>/etc/hosts
#
#
#
echo --- Instalando Heartbeat ----
#rpm    -Uvh    http://download.fedoraproject.org/pub/epel/6/x86_64/epel-release-6-
8.noarch.rpm
#yum install heartbeat
echo --- Configurando  Heartbeat ----
echo --- creando /etc/ha.d/authkeys ----

cat >> /etc/ha.d/authkeys << EOF
auth 2
2 sha1 twinn0r1tt@
EOF
chmod 600 /etc/ha.d/authkeys
echo --- creando /etc/ha.d/ha.cf ----

cat >> /etc/ha.d/ha.cf << EOF
logfile /var/log/ha-log
logfacility local0
keepalive 2
deadtime 30
initdead 120
bcast eth0
udpport 694
auto_failback on
node $h1
node $h2
EOF

echo --- creando /etc/ha.d/haresources ----

cat >> etc/ha.d/haresources << EOF
IPaddr::192.168.118.150/24/eth0
balanceador-1 IPaddr::192.168.3.15/24/eth2
EOF
service heartbeat start
chkconfig heartbeat on

#
#
#
```

```
echo --- Instalando HAproxy ----
yum install haproxy
cat >> /etc/haproxy/haproxy.cfg << EOF
blabla

EOF

chkconfig haproxy on
service haproxy start

echo --- Termino la instalacion  para Capa Balanceo ----
```

VII. Script instala web

```
#!/bin/bash
echo --- Configurando /etc/hosts ----
read -p "hostname Nodo1: " h1
read -p "IP Nodo1: " n1
read -p "hostname Nodo1: " h2
read -p "IP Nodo1: " n2
echo $h1        $n1 >>/etc/hosts
echo $h2        $n2 >>/etc/hosts
#
#
#
echo --- Instalando Apache ----
yum -y install httpd

echo --- Instalando Prerequisitos para Moodle ----

yum install -y php  php-mysql php-pdo php-gd php-soap php-xmlrpc php-xml  php-cli
php-mbstring php-pear mod_ssl

echo --- Instalando GlusterFS ----
wget -P /etc/yum.repos.d
http://download.gluster.org/pub/gluster/glusterfs/LATEST/CentOS/glusterfs-
epel.repo
yum -y install glusterfs glusterfs-fuse glusterfs-server glusterfs-client
mkdir  /mnt/datos

echo --- Termino la Instalacion para capa Web, Ahora configura GusterFS segun tus
necesidades ----
```

VIII. Script instala Galera

```
#!/bin/bash
echo --- Configurando /etc/hosts ----
read -p "hostname Nodo1: " h1
read -p "IP Nodo1: " n1
read -p "hostname Nodo1: " h2
read -p "IP Nodo1: " n2
echo $h1        $n1 >>/etc/hosts
echo $h2        $n2 >>/etc/hosts
#
#
#
touch /etc/yum.repos.d/maria.repo
cat >> /etc/yum.repos.d/maria.repo << EOF
[mariadb]
name = MariaDB
baseurl = http://yum.mariadb.org/5.5/centos6-amd64
gpgkey=https://yum.mariadb.org/RPM-GPG-KEY-MariaDB
gpgcheck=1
EOF
echo --- Instalando Galera-server ----


yum install -y MariaDB-Galera-server MariaDB-client galera
yum    install    -y    http://www.percona.com/downloads/percona-release/redhat/0.1-
3/percona-release-0.1-3.noarch.rpm
yum install -y  percona-toolkit percona-xtrabackup


echo --- Configurando Galera ----
service mysql start
mysql_secure_installation

cat >> /etc/my.cnf.d/server.cnf << EOF
[mariadb]
query_cache_size=0
binlog_format=ROW
default_storage_engine=innodb
innodb_autoinc_lock_mode=2
wsrep_provider=/usr/lib64/galera/libgalera_smm.so
wsrep_cluster_address=gcomm://$n2
wsrep_cluster_name=clusterbd
wsrep_node_address=$n1
wsrep_node_name=base1
wsrep_sst_method=rsync
wsrep_sst_auth=root:tw1nn0r1tt@
EOF

/etc/init.d/mysql stop
/etc/init.d/mysql bootstrap
```

IX. Resultados de pruebas con Benchmarck AB

Balanceo con 2 nodos, tipo de balanceo: Less-conection.

ab -n100 -c90 -C co=7m6l55cnfor1jcrh2292816nh4
http://192.168.118.150/moodle/login/index.php

Concurrency Level: 90.
Time taken for tests: 14.349 seconds.

Nodos	**Memoria RAM**	**Uso de CPU**
Balanceador	11.44%	1.50%
Web1	6.20%	6.45%
Web2	4.33%	5.56%
BD1	5.10%	2.11%
BD2	4.99%	2.40%

Tabla 37: Simulación de 100 peticiones y 90 conexiones simultáneas usando 2 nodos

10000 peticiones.
5 conexiones simultaneas a la web por petición.

Linea: ab -n10000 -c5 -C co=7m6l55cnfor1jcrh2292816nh4
http://192.168.118.150/moodle/login/index.php

Concurrency Level: 5.
Time taken for tests: 146.887 seconds.

Nodos	Memoria RAM	Uso de CPU
Balanceador	20.84%	16.80%
Web1	15.20%	60.15%
Web2	14.13%	54.38%
BD1	8.40%	25.71%
BD2	8.69%	25.42%

Tabla 38: Simulación de 10000 peticiones y 5 conexiones simultáneas usando 2 nodos

1000 peticiones.

500 conexiones simultaneas a la web por petición.

Linea: ab -n1000 -c500 -C co=7m6l55cnfor1jcrh2292816nh4

http://192.168.118.150/moodle/login/index.php

Concurrency Level: 500.

Time taken for tests: 145.412 seconds.

Nodos	Memoria RAM	Uso de CPU
Balanceador	16.45%	12.60%
Web1	13.12%	40.45%
Web2	13.33%	38.56%
BD1	8.12%	20.51%
BD2	8.49%	19.43%

Tabla 39: Simulación de 1000 peticiones y 500 conexiones simultáneas usando 2 nodos

10000 peticiones.

100 conexiones simultaneas a la web por petición.

Linea: ab -n10000 -c100 -C co=7m6l55cnfor1jcrh2292816nh4

http://192.168.118.150/moodle/login/index.php

Concurrency Level: 100.

Time taken for tests: 1463.543 seconds.

Nodos	**Memoria RAM**	**Uso de CPU**
Balanceador	40.44%	16.59%
Web1	26.20%	70.45%
Web2	24.33%	74.44%
BD1	9.10%	20.51%
BD2	9.99%	21.70%

Tabla 40: Simulación de 10000 peticiones y 100 conexiones simultáneas usando 2 nodos

Balanceo con 2 nodos, tipo de balanceo: Round Robin.

100 peticiones.

90 conexiones simultaneas a la web por petición.

Linea: ab -n100 -c90 -C co=7m6l55cnfor1jcrh2292816nh4

http://192.168.118.150/moodle/login/index.php

Concurrency Level: 90.

Time taken for tests: 14.670 seconds.

Nodos	**Memoria RAM**	**Uso de CPU**
Balanceador	12.44%	2.99%
Web1	6.11%	6.21%
Web2	5.95%	5.56%
BD1	5.22%	2.23%
BD2	5.30%	2.19%

Tabla 41: Simulación de 100 peticiones y 90 conexiones simultáneas usando 2 nodos

10000 peticiones.

5 conexiones simultaneas a la web por petición.

Linea: ab -n10000 -c5 -C co=7m6l55cnfor1jcrh2292816nh4

http://192.168.118.150/moodle/login/index.php

Concurrency Level: 5.

Time taken for tests: 251.159 seconds.

Nodos	**Memoria RAM**	**Uso de CPU**
Balanceador	21.02%	18.81%
Web1	11.18%	30.95%
Web2	11.13%	24.72%
BD1	5.19%	22.46%
BD2	5.81%	21.99%

Tabla 42: Simulación de 1000 peticiones y 5 conexiones simultáneas usando 2 nodos

1000 peticiones.

500 conexiones simultaneas a la web por petición.

Linea: ab -n1000 -c500 -C co=7m6l55cnfor1jcrh2292816nh4

http://192.168.118.150/moodle/login/index.php

Concurrency Level: 500.

Time taken for tests: 124.738 seconds.

Nodos	Memoria RAM	Uso de CPU
Balanceador	20.75%	15.23%
Web1	10.84%	22.22%
Web2	10.59%	21.84%
BD1	7.96%	18.39%
BD2	7.52%	17.55%

Tabla 43: Simulación de 1000 peticiones y 500 conexiones simultáneas usando 2 nodos

10000 peticiones.

100 conexiones simultaneas a la web por petición.

Linea: ab -n10000 -c100 -C co=7m6l55cnfor1jcrh2292816nh4

http://192.168.118.150/moodle/login/index.php

Concurrency Level: 100.

Time taken for tests: 1461.478 seconds.

Nodos	Memoria RAM	Uso de CPU
Balanceador	42.99%	15.32%
Web1	25.99%	69.95%
Web2	24.11%	68.98%
BD1	8.88%	19.99%
BD2	8.10%	20.87%

Tabla 44: Simulación de 10000 peticiones y 100 conexiones simultáneas usando 2 nodos

Prueba 2

Balanceo con 4 nodos, tipo de balanceo: Less-conection.

100 peticiones.

90 conexiones simultaneas a la web por petición.

Linea: ab -n100 -c90 -C co=7m6l55cnfor1jcrh2292816nh4
http://192.168.118.150/moodle/login/index.php

Concurrency Level: 90.
Time taken for tests: 9.465 seconds.

Nodos	Memoria RAM	Uso de CPU
Balanceador	20.39%	5.34%
Web1	3.41%	3.82%
Web2	3.95%	3.49%
Web3	3.23%	3.36%
Web4	3.58%	3.76%
BD1	8.63%	5.59%
BD2	8.67%	5.41%

Tabla 45: Simulación de 100 peticiones y 90 conexiones simultáneas usando 4 nodos

10000 peticiones.
5 conexiones simultaneas a la web por petición.

Linea: ab -n10000 -c5 -C co=7m6l55cnfor1jcrh2292816nh4
http://192.168.118.150/moodle/login/index.php

Concurrency Level: 5.
Time taken for tests: 113.940 seconds.

Nodos	Memoria RAM	Uso de CPU
Balanceador	25.02%	20.22%
Web1	5.24%	11.49%
Web2	5.45%	11.44%
Web3	5.55%	11.57%
Web4	5.34%	10.99%
BD1	9.79%	24.99%
BD2	9.99%	25.10%

Tabla 46: Simulación de 10000 peticiones y 5 conexiones simultáneas usando 4 nodos

1000 peticiones.

500 conexiones simultaneas a la web por petición.

Linea: ab -n1000 -c500 -C co=7m6l55cnfor1jcrh2292816nh4

http://192.168.118.150/moodle/login/index.php

Concurrency Level: 500.

Time taken for tests: 93.246 seconds.

Nodos	Memoria RAM	Uso de CPU
Balanceador	25.65%	19.72%
Web1	8.48%	19.44%
Web2	7.92%	19.79%
Web3	8.15%	19.56%
Web4	7.82%	19.23%
BD1	9.93%	25.29%
BD2	9.91%	24.79%

Tabla 47: Simulación de 1000 peticiones y 500 conexiones simultáneas usando 4 nodos

10000 peticiones.

100 conexiones simultaneas a la web por petición.

Linea: ab -n10000 -c100 -C co=7m6l55cnfor1jcrh2292816nh4
http://192.168.118.150/moodle/login/index.php

Concurrency Level: 100.
Time taken for tests: 933.095 seconds.

Nodos	Memoria RAM	Uso de CPU
Balanceador	48.31%	25.29%
Web1	20.01%	53.26%
Web2	20.05%	54.00%
Web3	19.26%	54.65%
Web4	19.43%	53.76%
BD1	10.88%	30.33%
BD2	10.10%	30.29%

Tabla 48: Simulación de 10000 peticiones y 100 conexiones simultáneas usando 4 nodos

Balanceo con 4 nodos, tipo de balanceo: Round robin.

100 peticiones.
90 conexiones simultaneas a la web por petición.

Linea: ab -n100 -c90 -C co=7m6l55cnfor1jcrh2292816nh4
http://192.168.118.150/moodle/login/index.php

Concurrency Level: 90.
Time taken for tests: 9.653 seconds.

Nodos	Memoria RAM	Uso de CPU
Balanceador	22.01%	5.03%
Web1	3.93%	3.25%
Web2	3.34%	3.30%
Web3	3.78%	3.33%
Web4	3.99%	3.34%
BD1	9.05%	5.09%
BD2	9.03%	5.05%

Tabla 49: Simulación de 100 peticiones y 90 conexiones simultáneas usando 4 nodos

10000 peticiones.

5 conexiones simultaneas a la web por petición.

Linea: ab -n10000 -c5 -C co=7m6l55cnfor1jcrh2292816nh4

http://192.168.118.150/moodle/login/index.php

Concurrency Level: 5.

Time taken for tests: 114.440 seconds.

Nodos	Memoria RAM	Uso de CPU
Balanceador	25.10%	20.28%
Web1	5.28%	11.52%
Web2	5.48%	11.50%
Web3	5.59%	11.53%
Web4	5.32%	11.53%
BD1	9.83%	25.15%
BD2	9.80%	25.13%

Tabla 50: Simulación de 10000 peticiones y 5 conexiones simultáneas usando 4 nodos

1000 peticiones.

500 conexiones simultaneas a la web por petición.

Linea: ab -n1000 -c500 -C co=7m6l55cnfor1jcrh2292816nh4
http://192.168.118.150/moodle/login/index.php

Concurrency Level: 500.
Time taken for tests: 96.050 seconds.

Nodos	Memoria RAM	Uso de CPU
Balanceador	25.98%	20.11%
Web1	8.50%	19.58%
Web2	8.02%	19.88%
Web3	8.08%	19.80%
Web4	7.99%	19.77%
BD1	9.96%	25.25%
BD2	9.90%	24.30%

Tabla 51: Simulación de 1000 peticiones y 500 conexiones simultáneas usando 4 nodos

10000 peticiones.
100 conexiones simultaneas a la web por petición.

Linea: ab -n10000 -c100 -C co=7m6l55cnfor1jcrh2292816nh4
http://192.168.118.150/moodle/login/index.php

Concurrency Level: 100.
Time taken for tests: 974.499 seconds.

Nodos	Memoria RAM	Uso de CPU
Balanceador	50.91%	30.09%
Web1	20.89%	54.56%
Web2	20.90%	54.60%
Web3	20.91%	54.66%
Web4	20.91%	53.66%
BD1	10.80%	30.99%
BD2	10.59%	30.89%

Tabla 52: Simulación de 10000 peticiones y 100 conexiones simultáneas usando 4 nodos