UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE ESTUDIOS SUPERIORES ACATLÁN

"ALTA DISPONIBILIDAD EN BASES DE DATOS PARA EL DEPARTAMENTO DE SERVIDORES DE LA DGSCA"

T E S I S

QUE PARA OBTENER EL TÍTULO DE:

LICENCIADA EN MATEMATICAS APLICADAS Y COMPUTACIÓN

P R E S E N T A :

RAQUEL AMPARO SERAFÍN SALINAS

ASESORA:

M. C. MARÍA DEL CARMEN VILLAR PATIÑO

ACATLÁN, ESTADO DE MÉXICO 2008

MI MAS SINCERO AGRADECIMIENTO Y AMOR …

Un objetivo más en mi vida cumplido papá y mamá. Qué satisfacción más grande en la vida que cumplir lo que uno se propone, ¿no es lo que me enseñaron? "No importa lo que pase, no importa lo que venga, cumple lo que te propones, aquí tienes a tus padres" decía mi papá. Jamás creí vivirlo en carne propia, todo era felicidad, los obstáculos que venían en camino por más difíciles que aparentaran ser, en realidad no lo eran, porque ahí estaban mis padres, siempre ahí; después y tan rápido, todo cambia.

La vida golpea duro, aún así, presento el siguiente trabajo. Tienes mucho mérito papá, gracias por el apoyo a pesar de estar destrozado por dentro, siempre fuiste y has sido fuerte, por ti, por mis hermanos y la memoria de mi madre es por lo que aún estoy de pie y presente para recibir mi título profesional, no puede ser de otra forma.

A mi alrededor hubieron amigos que me animaron cuando más lo necesité, gracias Salvador Cruz, sin darte cuenta arrancabas muchas sonrisas de mi rostro cuando parecía que sonreír estaba fuera de mi ser. A ti mi querido Jorch, nunca te olvidaré, fuiste el mejor soporte que Dios me pudo haber mandado, gracias de todo corazón. Mi muy querida Mago, jamás nos abandonaste, gracias por el apoyo. Mi personita especial, Carlos Fernando, gracias corazón.

Y a la mejor de todos, mi madre, mi angelito de la guarda, gracias por todo mami.

Con amor e infinito agradecimiento, a mis padres está dedicada la presente tesis.

Tabla de contenido

INTRODUCCIÓN

En el ámbito de la administración de servidores siempre es necesario el desarrollo de nuevas propuestas que den solución a los distintos problemas que se presentan al administrar servidores de bases de datos, páginas Web, correo electrónico, etc.; o simplemente para facilitar la labor del administrador.

Este trabajo, presenta una propuesta a las necesidades del departamento de Administración de Servidores de la Dirección General de Servicios de Cómputo Académico para una de las aplicaciones que administra: "Tiendas electrónicas UNAM". El responsable de ésta aplicación solicita "Alta Disponibilidad"; es importante tomar en cuenta que dicha aplicación Web trabaja con una base de datos.

El objetivo es maximizar el tiempo disponible de la aplicación en línea haciendo uso de paquetes o programas de código abierto y los recursos de hardware y software disponibles en el departamento.

Restricciones fuera del ambiente de la Tecnología de la Información hace que éste reto sea casi imposible de cumplir. Estas restricciones incluyen limitaciones de presupuesto, fallas en los componentes, código de software mal escrito, error humano, diseños erróneos, etc. Estos son los factores que trabajan en contra del ideal del 100% de disponibilidad.

Conforme se desarrolla el trabajo se presenta la información necesaria para la compresión de la propuesta de alta disponibilidad para cumplir el objetivo.

El primer capítulo da una introducción al papel que desempeñan los Servidores Web, para cubrir éste punto se da una explicación de la arquitectura cliente-servidor. Se puntualizan los tipos de servidores que existen mencionando una parte muy importante que es el Hospedaje de páginas Web, se explican las características de los sistemas operativos Linux y Unix dado que el sistema operativo es un elemento indispensable que define el funcionamiento del servidor.

El segundo capítulo abarca Alta Disponibilidad y Balanceo de Carga. Se explica a detalle el concepto de UltraMonkey, proyecto que define un sistema de Alta Disponibilidad. Una vez comprendido el concepto, se listan las topologías con las que funciona, entre ellas, la correspondiente a la propuesta.

Debido a que la aplicación de "Tiendas electrónicas UNAM" trabaja con una base de datos, hay que darle un trato especial para brindar alta disponibilidad a la aplicación completa, por tanto el tercer capítulo abarca lo correspondiente a la replicación de datos.

En el último capítulo se documenta el punto a tratar, la aplicación "Tiendas electrónicas UNAM", el hardware disponible y la estructura de la propuesta del sistema de Alta Disponibilidad en Bases de Datos para el Departamento de Servidores de la DGSCA, desde el sistema operativo y rpm's necesarios hasta la configuración en cada máquina, archivos de configuración y la red para el correcto funcionamiento.

CAPÍTULO I SERVIDORES WEB

1.1 LA WEB E INTERNET

El origen de Internet, de "la Red" o también llamada "Red de redes", se remonta a finales de la década de los 60, cuando se logra interconectar un pequeño número de máquinas. En estos más de 40 años, de este reducido número de máquinas ha pasado a millones en todo el mundo y su evolución e integración ha sido tal que ha logrado convertirse en algo cotidiano para un grupo de personas cada vez mayor.

El nacimiento de la Red comienza hacia 1964 cuando las autoridades americanas comenzaron a plantearse el problema estratégico de las comunicaciones después de una posible guerra nuclear. La Corporación RAND fue la encargada de estudiar las soluciones a tal problema Se buscaba un sistema que pudiera transmitir órdenes a todos los puntos estratégicos: bases militares, ciudades, estados, etc.

La ARPA (*Advanced Research Projects Agency*) del Pentágono decidió entonces realizar un proyecto más ambicioso: consistía en poner, un conjunto de "súper máquinas" de alta velocidad.

En diciembre de 1969 existían cuatro nodos que dieron paso al nacimiento de la red ARPAnet, en 1971 ya existían quince nodos, al año siguiente crecieron a treinta y siete. En el segundo año de la operación se empezaron a gestionar los primeros sistemas de correo electrónico. Comenzaron a nacer rápidamente las listas de correo, aparecieron las primeras conferencias en línea, que tenían lugar mientras la persona se encontraba realizando una conexión con la red.

El TCP (*Transmisión Control Protocol*) se encargaba de convertir en paquetes los mensajes de la fuente y volverlos a montar en el lugar de destino. IP (*Internet Protocol*) administraba el camino y el direccionamiento que llevaban los paquetes, de tal forma que pasaran por los múltiples nodos y redes de máquinas intermedias correspondientes.

En 1977, el TCP/IP era utilizado por otras redes para enlazarse a ARPAnet. Cualquier máquina podía conectarse a la red. Sólo necesitaba comprender y hablar el lenguaje empleado para el intercambio de paquetes.

En poco tiempo, los servicios que ofrece Internet se han convertido en un elemento más de la vida cotidiana de cualquier persona. Ahora, Internet es la red de cómputo más grande en el mundo, consiste en miles de redes conectadas que se comunican para compartir información.

World Wide Web o también llamado Web es una de las herramientas con las que cuenta el usuario para la búsqueda y difusión de datos en Internet, creada por el desarrollo de la idea de hipertexto, se ha convertido en el servicio más importante de Internet. En la Web existen una serie de documentos que pueden ser recuperados y examinados por cualquiera que tenga acceso a Internet en cualquier parte del mundo. Por su carácter documental, un servidor Web puede ofrecer información de carácter didáctico, técnico, científico, etc.[*1]

1.1.1 Conexión Web

Un protocolo de comunicación es un estándar de comunicaciones que muchas computadoras pueden entender. En el centro de todos los protocolos en Internet está TCP/IP. Este tipo es el protocolo estándar básico que permite la transmisión de paquetes de datos mediante Internet. Los protocolos usados comúnmente en Internet son:

[*1] Recopilación de información del libro [1], capítulos I y II.

FTP (File Transfer Protocol), usado para transferencia de archivos, facilita ciertas medidas de seguridad que pueden restringir el acceso a archivos y servidores particulares.

HTTP (*HiperText Transfer Protocol*), protocolo de transferencia por hipertexto, diseñado para transferir documentos a mayor velocidad.

Cuando se realiza una conexión Web, el navegador envía instrucciones al programa de comunicaciones para contactar con una computadora específica en la Web y recobrar un documento o página Web concreta. El programa de comunicaciones permite al navegador enviar la solicitud usando el protocolo TCP/IP, este protocolo divide la petición, y la respuesta a ella en pequeños paquetes de datos que se mandan a través de Internet.

Cuando se examina en las páginas Web mediante un navegador en una computadora cliente, normalmente hay desplazamiento entre ellas y múltiples sitios Web. El nombre de la ubicación que aloja las páginas se denomina dominio. Para configurar un sitio Web es necesario registrar un nombre de dominio único asociado al sitio, en México, los nombres de dominio se asignan por medio de NicMexico con extensiones .mx, .com.mx, .edu.mx, .net.mx, .org.mx, .gob.mx.

Una vez que la computadora cliente se comunica con un dominio, el nombre debe transformarse en un esquema numérico llamado dirección IP (los nombres son "acordados" para las direcciones IP que usan el sistema de nombres de dominio o DNS). Existen los servidores de nombres de dominio que alojan tablas llenas de nombres de dominio y sus correspondientes direcciones IP. El DNS y los nombres abastecen de clientes al servidor correcto con la página Web apropiada para poder realizar conexiones Web satisfactorias.

Más adelante, en la arquitectura cliente servidor, se explica más a detalle éste procedimiento. Por el momento sólo quiero hacer mención de éste y dar un panorama general.

1.1.2 HTML e hipertexto

El lenguaje de marcado de hipertexto (HTML, *HyperText Markup Language*) define el formato de un documento Web y permite incluir enlaces de hipertexto en el documento, éste es el lenguaje de marcado específico de la Web. El HTML se ha convertido en uno de los formatos más populares y fáciles de aprender que existen para la elaboración de documentos para Web.

A finales de los años sesenta, fue introducido un concepto que se apoya en las bases de la Web y sus conexiones entre documentos o páginas, "hipertexto". Este concepto surge de la idea de crear una nueva forma de explorar la información, de compartir trabajos de investigación, artículos, etc., a través de Internet. La idea fue visualizar un sistema donde un documento podría enlazarse con otros, permitiendo de esta forma encontrar fácilmente más información y además relacionada con el sólo hecho de seguir un enlace de un documento en la red a otro.

El hipertexto consta de un hiperenlace que aparece en pantalla como una palabra, ícono o gráfico enfatizado.

Es posible crear páginas Web utilizando únicamente HTML, sin embargo, en la actualidad existen una gran variedad de herramientas y lenguajes de programación que en conjunto con HTML permiten crear páginas Web complejas que integran audio, video, animaciones, interactúan con bases de datos, etc.[*2]

[*2] Conceptos e información obtenidos del libro [1], p. 59, 60, 751, 752

1.2 SISTEMAS OPERATIVOS

El sistema operativo es en sí mismo un programa complejo e importante. Cuando se conecta una máquina se carga parte del sistema operativo en la memoria y se ejecuta. El sistema operativo llama a la máquina y hace que reconozca la CPU, memoria, unidades de disco y cualquier otro dispositivo conectado a ella como el teclado, ratón, etc., verificando que no existen errores de conexión y que los dispositivos sean reconocidos y trabajen correctamente. Sus funciones principales consisten en el control de la ejecución de los programas, administración de periféricos, administración de permisos y de usuarios, control de concurrencia, errores y seguridad, administración de memoria.

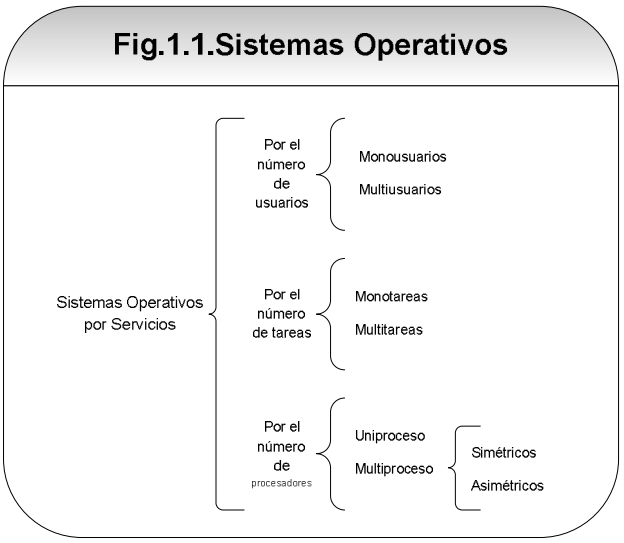
Un sistema operativo es un programa o conjunto de programas que actúan como intermediario entre el usuario y el hardware de la máquina, administrando los recursos del sistema y optimizando su uso.

1.2.1 Clasificación de los sistemas operativos

Según la perspectiva con la que se observen los sistemas operativos, pueden realizarse múltiples clasificaciones. Me parece importante resaltar la clasificación de los sistemas operativos en cuanto a servicios ofrecidos y estructura interna, por ello, a continuación muestro dicha clasificación de acuerdo con Laura Raya[2] con el objetivo de mostrar más a detalle su funcionamiento y más adelante comprender los conceptos que se mencionan para Unix y Linux.

1.2.1.1 Sistemas operativos por servicios ofrecidos

En esta clasificación se tiene en cuenta la visión del usuario final y se comprende fácilmente con el siguiente cuadro sinóptico **Fig.1.1.**



Los **sistemas operativos monousuario** son aquéllos que únicamente soportan un usuario a la vez, sin importar las características de la máquina sobre la que está montado el sistema.

Los **sistemas operativos multiusuario** son capaces de dar servicio a más de un usuario a la vez, ya sea por medio de varias terminales conectadas a la computadora o por medio de sesiones remotas en una red de comunicaciones, también independientemente de la plataforma hardware sobre la que esté montado.

Los **sistemas operativos monotarea** son aquellos que sólo permiten una tarea a la vez por usuario. Puede darse el caso de un sistema multiusuario y monotarea, en el cual se admiten varios usuarios al mismo tiempo pero cada uno de ellos puede estar haciendo solo una tarea a la vez.

Un **sistema operativo multitarea** es aquel que le permite al usuario estar realizando varios trabajos al mismo tiempo. Es común encontrar en ellos interfaces gráficas orientadas al uso de menús y al ratón, lo que permite un rápido intercambio entre las tareas para el usuario, mejorando su productividad. En los sistemas multitarea de tiempo compartido, cada tarea recibe la atención del microprocesador durante una fracción de segundo. Un sistema operativo multitarea puede estar editando el código fuente de un programa durante su depuración mientras compila otro programa, a la vez que está recibiendo correo electrónico en un proceso en background.

Los **sistemas operativos monoproceso** son los que solamente permiten realizar un proceso a la vez. Sin embargo, permiten simular la multitarea haciendo que el sistema realice una tarea rotatoria con intercambio muy rápido

Los **sistemas operativos Multiproceso** permiten realizar varios procesos simultáneamente y, por tanto, son capaces de ejecutar varias tareas al mismo tiempo.

Dentro de los sistemas multiproceso, se encuentran los sistemas **simétricos**, que son los que distribuyen la carga de procesamiento por igual entre todos los procesadores existentes. Sin embargo, los sistemas multiproceso **asimétricos**, asignan una tarea por procesador existente, según su prioridad, y el resto de tareas de baja prioridad se ejecutan en un único procesador.

1.2.1.2 Sistemas operativos por su estructura interna

Esta clasificación comprende cómo se diseñan los sistemas a la hora de ser creados. Hay que tener en cuenta que, en la mayoría de los casos, estas concepciones no se deben entender aisladas, en ocasiones existe interrelación. A continuación se describen las distintas estructuras que presentan los actuales sistemas operativos para satisfacer las necesidades que de ellos se quieren obtener.

Estructura Monolítica

Es la estructura utilizada en los primeros sistemas operativos en la que todas las funciones se implementaban en el kernel. Puede decirse que su estructura consiste en que no existe una estructura como tal. El sistema operativo está constituido por un único programa compuesto de multitud de rutinas interrelacionadas entre sí, de forma que cada una de ellas pueda llamar a cualquier otra. Generalmente están hechos a medida, por lo que son eficientes y rápidos en su ejecución y gestión, pero por lo mismo carecen de flexibilidad para soportar diferentes ambientes de trabajo o tipos de aplicaciones.

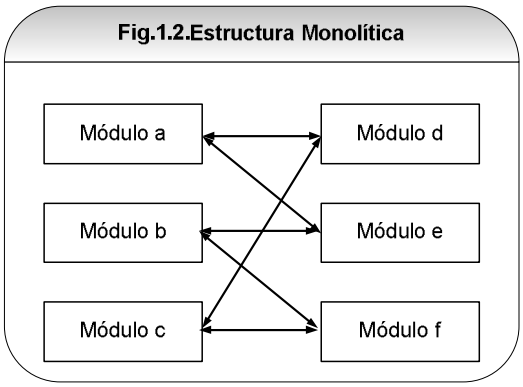


Fig.1.2.Estructura Monolítica

Fig.1.2. estructura monolítica[*3], muestra la interrelación de rutinas por las que está conformado el sistema operativo.

Estructura por capas

A medida que los sistemas operativos fueron creciendo, fue siendo necesaria una mayor estructuración. Este diseño corresponde con una estructura jerárquica que se divide en distintos niveles, donde una parte del sistema contiene subpartes y estos organizados en forma de niveles de manera que cada una de ellas estuviera perfectamente definida y con un claro objetivo para con el resto de los elementos. Se puede pensar también en estos sistemas como si fueran `multicapa'. Multics y Unix caen en esa categoría.

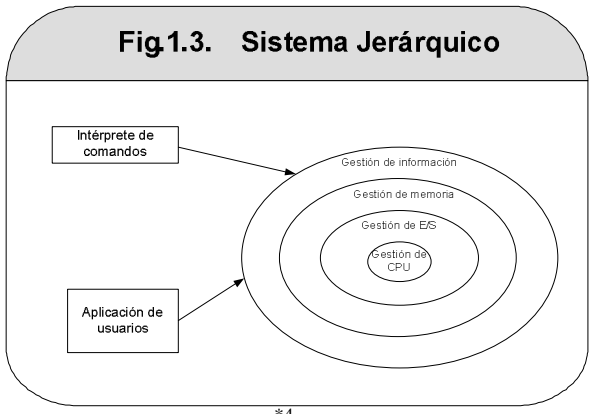


Fig.1.3. Sistema jerárquico en forma de anillos[*4], muestra la estructura por capas o jerárquica por niveles.

1.2.2 Unix

UNIX es un sistema operativo cuyos comienzos se remontan a principios de los años setenta aproximadamente. No surgió como un producto comercial, sino como un proyecto personal de Ken Thompson y Dennis Ritchie, quienes trabajaban en los Laboratorios Bell. La idea básica que inspiró su nacimiento fue la de crear un entorno de trabajo simple y agradable para el desarrollo de aplicaciones el proyecto se denominó UNIX. Dennis Ritchie quien ya había trabajado anteriormente en un proyecto llamado MULTICS (proyecto que por sus objetivos demasiado ambiciosos para la época fracasó) tuvo mucha influencia en el nuevo sistema operativo. Como ejemplos de esa influencia puedo mencionar la organización básica del sistema de archivos, la idea del intérprete de órdenes (shell) como proceso de usuario e incluso el propio nombre UNIX deriva de MULTICS. Dotaron al nuevo sistema operativo de la capacidad de soportar varios programas cargados en memoria en un mismo instante. También aportaron al nuevo sistema la capacidad de tiempo compartido, es decir, tiempo total del procesador repartido entre todas las aplicaciones en rodajas o cuantos de tiempo, mejorando con ello los tiempos de respuesta. De este modo, se puede tener a varias personas conectadas al mismo tiempo, y desde distintas terminales, a la misma máquina.

En 1973 el sistema operativo fue reescrito en lenguaje C en su mayor parte. C es un lenguaje de alto nivel (las versiones anteriores del sistema operativo habían sido escritas en ensamblador), lo que propició que el sistema tuviera una gran aceptación por parte de los nuevos usuarios. El número de instalaciones en Bell Laboratories creció hasta 25, aproximadamente, y su uso también se difundió gradualmente a unas cuantas universidades con propósitos educacionales.

[*3] Imagen disponible en Internet vía Web URL: html.rincondelvago.com/teleinformática_5.htm
[*4] Imagen disponible en Internet vía Web URL: http://www.geocities.com/bayron_g/tipos_de_sistemas_operativos.htm

Desde sus orígenes a la actualidad, UNIX ha sufrido multitud de modificaciones. Se le han ido añadiendo nuevas posibilidades, como la capacidad de conexión en red, entornos de ventanas, el soporte para diferentes arquitecturas, por ejemplo Sun Microsystems lo comercializa para sus computadoras con el nombre de Solaris, IBM como AIX, HP como HP-UX, Silicon Graphics como IRIX, etc. También, y debido a la evolución del hardware de las computadoras personales, existen versiones de UNIX para PC, de las cuales conviene resaltar aquellas que son de libre distribución, como Linux, 386BSD y FreeBSD.[*5]

Respecto a lo anterior, Sebastián Sánchez[3] menciona: "los sistemas UNIX actuales no se reducen a la Versión 8, System V o BSD, sino que la mayoría de los fabricantes de micro y minicomputadoras ofrecen su UNIX particular".

Unix:

- Es multiusuario
- Es multitarea
- Tiempo compartido
- Cuenta con memoria virtual
- Fue desarrollado en lenguaje de alto nivel C
- Es un sistema operativo optimizado para hardware específico, lo que ofrece rendimiento y fiabilidad
- Cuenta con soporte del fabricante para su SO y hardware.

1.2.3 Linux

Linux es un sistema operativo de distribución libre desarrollado inicialmente por Linus Torvals en la universidad de Helsinki en Finlandia. Con la aparición de máquinas personales potentes aparece Linux. Inicialmente fue sólo un desarrollo llevado a cabo por Linus Torvalds inspirado en Minix, pequeño sistema UNIX desarrollado por Andrew S. Tanenbaum. Sus primeros comentarios tenían que ver con el desarrollo de un sistema operativo académico que superara los estándares de Minix. Después de varias versiones intermedias, Linus incrementó el número de la versión y pasó directamente a la versión 0.95 para reflejar deseos de pasar su proyecto a una versión "oficial" (el número de versión 1.0 supone que está en su mayoría libre de errores), esto en el año 1992.

Una comunidad de programadores distribuidos por Internet han ayudado en el desarrollo, depuración y distribución de este sistema operativo. La mayoría del software disponible en Linux ha sido desarrollado por el proyecto GNU de la Free Software Foundation de Cambridge (Massachusetts). De hecho, el kernel es desarrollado y liberado bajo la licencia publica general de GNU y su código fuente es de libre distribución y disponible para cualquiera.

Actualmente Linux es un UNIX en toda regla, capaz de ejecutar X-Window (sistema de ventanas portable que se ejecuta de forma transparente en red en diferentes plataformas y sistemas operativos), TCP/IP, correo electrónico, etc. La mayoría de los paquetes software de libre distribución han sido portados a Linux y cada vez son más las aplicaciones comerciales disponibles. Su robustez y el hecho de ser gratuito ha propiciado que Linux lo empleen como herramienta de desarrollo desde entidades de investigación hasta en productoras de cine.[*6]

[*5] Recopilación de información de los libros [2] y [3], p. XIII y capítulo I

[*6] La página oficial de Linux es muy usada para obtener información relevante y actualizada, referencia [13] , libro [3] p. 5, 6.

Linux:

- Es gratuito
- No hay dependencia con fabricantes únicos
- Es de código abierto
- No existe soporte específico de acuerdo a las necesidades
- La instalación de hardware se suele complicar un poco
- Multiusuario
- Multitarea
- Tiempo compartido
- Memoria virtual
- Se cuenta con libertad de elección entre distribuciones

1.2.3.1 Las distintas distribuciones

Cuando Linus Torvalds desarrolló Linux, el sistema operativo consistía básicamente de un kernel y herramientas GNU. Con la ayuda de otros programadores, Linus agregó cada vez más y más herramientas y aplicaciones a Linux. Con el tiempo, individuos, estudiantes universitarios y compañías comenzaron a distribuir Linux con sus propios packages (archivo directorio comprimido o no que contiene archivos para un programa particular que se quiere instalar) aunados al kernel desarrollado por Linus. Es aquí donde surge el concepto de "distribución".

Se puede descargar Linux de distintas compañías e individuos. Existen distribuciones de todo tipo y para cualquier tipo de computadora.

A continuación mencionaré tres de las distribuciones con las que he tenido contacto y considero estables para trabajar como sistema operativo de un servidor:

Red Hat Linux

Red Hat es reconocido como líder en iniciativa de soluciones que toma gran ventaja de las innovaciones provenientes del modelo de open source (código abierto, software libre que pone a disposición de la gente el código fuente del programa para que pueda ser estudiado, modificado y distribuido).

Con Red Hat, empresas de hardware y software cuentan con una plataforma estándar sobre la cual certificar su tecnología. Asegura la escalabilidad y seguridad necesaria del software de código abierto. Hace posible soportar aplicaciones de misión crítica y en conjunto, brinda bastantes beneficios en un ambiente empresarial donde el mejoramiento de costos es crítico.

En Noviembre del 2007, la revista CIO Insight anunció los resultados de su informe anual Vendor Value Survey, donde muestra a Red Hat como la empresa tecnológica mejor valorada por sus clientes, lo que es el mayor vendedor mundial de software libre. Y una puntuación de 97 sobre 100 de clientes que desean continuar haciendo negocios con Red Hat.[*7]

Red Hat brinda software completamente seguro, estable y con soporte. Ofrece integración en su tecnología y todas las piezas adecuadas para ello, de tal forma que desarrolla, integra, actualiza, maneja y da soporte a sus clientes. Ofrece un grande rango de infraestructura de middleware y manejo de soluciones[16].

[*7] Referencia de las puntuaciones http://www.redhat.com/promo/vendor

[16] Sitio oficial que brinda información de Red Hat distribución de Linux, compañía y subsidiarios.

SUSE Linux

SUSE Linux Enterprise de Novell es una plataforma completa de código abierto, ofrece confiabilidad para aplicaciones de misión crítica, virtualización, seguridad, manejo general de herramientas entre otras.

El código abierto (open source) está cambiando los modelos económicos y creando empresas dinámicas. Novell pretende encabezar este concepto con SUSE Linux Enterprise. Su colaboración en la comunidad de código abierto ha sido un factor importante para su éxito. Algunas de las más importantes contribuciones que Novell ha hecho a la comunidad es el soporte del proyecto openSUSE, provee de drivers libres de código abierto y promueven documentación a múltiples niveles. Aunado a lo anterior, Novell patrocina y contribuye al proyecto GNOME.

El proyecto openSUSE tiene más de 64,000 usuarios registrados y miles de personas que "testean" openSUSE y que retroalimentan el proyecto. La distribución openSUSE es la base de la plataforma SUSE Linux Enterprise de Novell y una rigurosa comunidad junto con Novell aseguran que la plataforma está lista y funciona correctamente.[*8]

Debian

El proyecto Debian es una asociación de individuos con algo en común, crear un sistema operativo libre. Este sistema operativo es llamado Debian GNU/Linux o simplemente Debian.

Debian cuenta con cerca de 18733 packages (software precompilado con cierto formato que proporciona fácil instalación en la máquina), todos estos libres. Este sistema operativo brinda un avanzado manejo de herramientas que permiten una fácil instalación y mantenimiento del sistema.

La base es el kernel, aunado a éste se encuentran todas las herramientas básicas y le siguen todo el software que se quiera correr en la máquina.

1.2.4. Sistemas operativos para servidores Web

Como hemos visto a lo largo de este tema, existen características importantes que definen un sistema operativo, y podemos mencionar dos grupos bien diferenciados: aquellos basados en Windows (NT, 2000, 2003) y los basados en Unix (HP-UX, Solaris, AIX, Linux).

Unix designa el núcleo de un sistema operativo multiusuario y multitarea. Orientado en primera instancia a terminales de caracteres, actualmente dispone de la interfaz gráfica X-Window. Esto ha simplificado mucho el uso para los no especialistas. Es ideal para trabajar como servidor. Se orienta en la dirección contraria a la tendencia de hacer invisible al usuario el sistema operativo, permitiendo el uso de todas las bibliotecas, llamadas al sistema y herramientas internas, aunque su uso requiere un alto nivel de especialización.

Unix/Linux cuentan con muchísimas aplicaciones que generan un ambiente estable y seguro. Por ejemplo, como servidor Web emplean habitualmente Apache, éste dispone de mayores opciones de configuración de forma relativamente sencilla.

Recordemos que los sistemas operativos Windows y Unix tienen costo debido a que es proveniente de marca registrada. Contrario a ellos se encuentra Linux, que representa un sistema operativo Unix de bajo costo, sin complicaciones de licencias y corre en una gran variedad de plataformas de hardware.

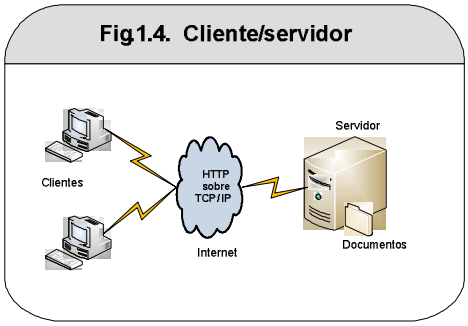
[*8] La pàgina oficial de suse brinda información verídica y reciente de la distribución [14]

En el entorno informático actual, una máquina Windows, Unix o Linux puede ser un cliente. Cualquiera de estas plataformas actúa como servidor e incluso puede hacerlo como ambas, cliente y servidor simultáneamente. Esta doble función es posible debido a las capacidades multitarea de los modernos sistemas operativos.

Ambos grupos trabajan de forma distinta debido a su estructura y los servicios ofrecidos; y la elección entre un sistema operativo u otro depende de las necesidades específicas del administrador lo cual de alguna forma condiciona su funcionamiento y crecimiento.

1.3 ARQUITECTURA CLIENTE/SERVIDOR

Cliente/servidor es una tecnología, clientes y servidores son entidades lógicas autónomas que trabajan juntas en una red para cumplir una tarea. Hace referencia a una relación cooperativa entre dos o más entidades. Su funcionamiento en la mayoría de los casos implica que el cliente haga peticiones requiriendo recursos y servicios que el servidor le proporciona. Ambos procesos pueden existir en máquinas separadas que se comunican a través de una red. La arquitectura cliente/servidor es uno de los pilares que fundamentan la mayoría de las aplicaciones que se encuentran en Internet. **Fig.1.4.** A grandes rasgos, podríamos decir que lo que se utiliza para navegar por Internet son programas "cliente" que acceden a otros programas que son servidores.



La mayoría de las soluciones cliente/servidor son arquitecturas de dos capas. Esto significa que la aplicación lógica es dividida entre las aplicaciones del cliente y la base de datos. Todo sistema cliente/servidor se distingue por las siguientes características:

Servicio

La arquitectura cliente/servidor es, ante todo, una relación entre procesos que se ejecutan en máquinas independientes una de la otra. El proceso servidor es un proveedor de servicios; el cliente los consume. En esencia, la tecnología cliente/servidor provee una clara separación de funciones con base en la idea de servicio.

Recursos compartidos

Un servidor puede servir a varios clientes al mismo tiempo y regular su acceso a recursos compartidos.

Protocolos asimétricos

Existe una relación de muchos a uno entre varios clientes y un servidor. Los clientes siempre empiezan el diálogo al solicitar un servicio; los servidores esperan de modo pasivo a que les lleguen solicitudes de los clientes. Obsérvese que, en algunos casos, un cliente podría pasar una referencia de retrollamada a un objeto cuando solicita un servicio, lo cual permite al servidor devolver la llamada (call back) al cliente. Así, el cliente se convierte en servidor.

Transparencia de ubicación

Un servidor es un proceso que puede residir en la misma máquina que el cliente, o en otra, en la red. Normalmente, el software cliente/servidor oculta a los clientes la ubicación del servidor redireccionando las solicitudes de servicio que le son requeridas. Un programa puede ser cliente, servidor, o las dos cosas.

Intercambios basados en mensajes

Clientes y servidores son sistemas acoplados sin grandes restricciones que interactúan mediante un mecanismo de intercambio de mensajes; así, éstos se convierten en el mecanismo de entrega para las solicitudes y respuestas de servicio.

Encapsulado de servicios

El servidor es un "especialista". A través de un mensaje se le indica cuál es el servicio que se le solicita, y luego depende de él la forma en que se satisface la solicitud. Los servidores pueden actualizarse sin afectar a los clientes siempre y cuando la interfaz de mensajes publicada no cambie.

Escalabilidad

Los sistemas cliente/servidor pueden actualizarse horizontal o verticalmente. El escalamiento horizontal implica que al agregar o quitar estaciones de trabajo clientes sólo se produce un pequeño efecto en el desempeño. El escalamiento vertical significa migrar a una máquina servidor más grande y rápida, o distribuir la carga de procesamiento entre varios servidores.

Integridad

El código y la información del servidor se administran de manera central, lo que da como resultado un mantenimiento más barato y el resguardo de la integridad de la información compartida. Al mismo tiempo, los clientes permanecen personales e independientes.

Las características de la tecnología cliente/servidor arriba descritas permiten distribuir sin problemas la inteligencia a lo largo de la red. Asimismo, brindan el marco para diseñar aplicaciones de red sin grandes restricciones.[*9]

1.3.1 Infraestructura del software cliente/servidor

Los tres bloques básicos de la tecnología cliente/servidor son: el cliente, el servidor y el middleware, que se encarga de unirlos. Con la finalidad de tener una mejor comprensión del tema, enseguida se describen los siguientes conceptos:

Interfaz del usuario

La interfaz es la conexión establecida entre la aplicación y el usuario final. El usuario final se puede comunicar con la aplicación a través de comandos y menús. Algunas aplicaciones utilizan Interfaces de Usuario Gráficas (*Graphical User Interfaces* GUI), que toman ventaja de ventanas predefinidas e íconos para proporcionar una apariencia común y reducir el tiempo de desarrollo o una interfaz de usuario orientada a objetos y que accede a los servicios distribuidos dondequiera que se encuentren.

[*9] Una introducción a la arquitectura Cliente/Servidor, fuente de información del libro [1] capítulo II, libro [6] capítulo I, libro [4] capítulo IV y p. 154.

Cliente

Es una máquina que accede a recursos y servicios brindados por otro llamado Servidor, generalmente en forma remota. El cliente hace peticiones al servidor y éste puede interactuar con uno o varios servidores para completar su trabajo.

Servidor

Ejecuta su proceso a la llegada de una petición. El servidor siempre está en espera de esta, ofreciendo servicios a las máquinas cliente. Esos servicios pueden ser brindados directamente por el servidor o indirectamente por un proceso esclavo generado por cada petición del cliente. Un servidor tiene una función específica, ejecuta un conjunto de transacciones predefinidas, relacionadas funcionalmente.

Middleware

Es un término que abarca a todo el software distribuido necesario para el soporte de interacciones entre Clientes y Servidores. Es el enlace que permite que un cliente obtenga un servicio de un servidor. Este se inicia en la parte del cliente que se emplea para invocar un servicio real; esto pertenece a los dominios del servidor. Existen dos tipos de middleware: Middleware general, permite la impresión de documentos remotos, manejos de transacciones, autenticación de usuarios, etc; middleware de servicios específicos, generalmente trabajan orientados a mensajes, trabaja una sola transacción a la vez.

En la **Fig.1.5** se dan más detalles acerca de lo que hay dentro de cada uno de esos bloques básicos. En pocas palabras, lo que se observa en la figura es la infraestructura del software cliente/servidor.[*10]

Veamos cada una de las piezas:

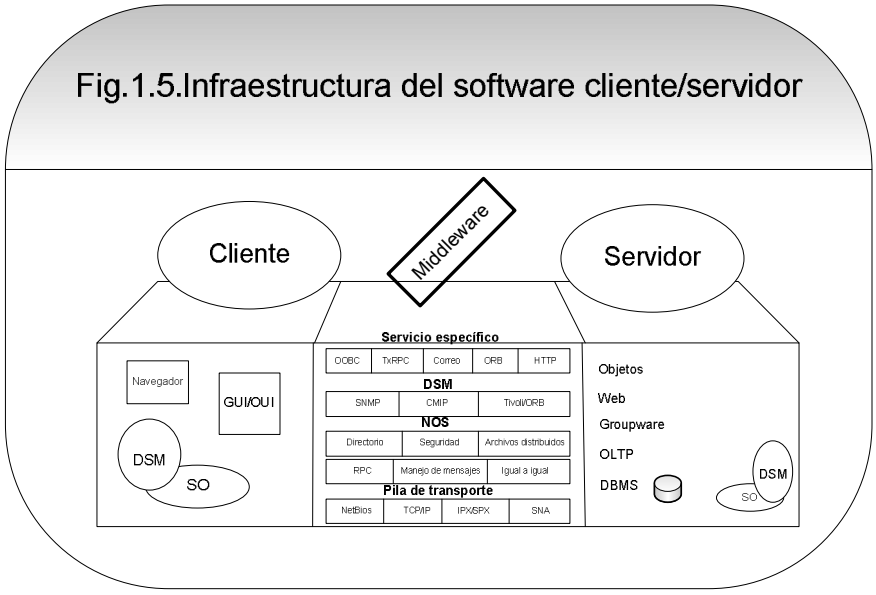


Fig.1.5.Infraestructura del software cliente/servidor

[*10] Robert Orfali hace una explicación completa de los bloques básicos de la tecnología Cliente/Servidor libro [6], capítulo VI.

El bloque básico del cliente

Ejecuta el lado cliente de la aplicación. Corre sobre un sistema operativo que proporciona una interfaz gráfica de usuario (GUI) o una interfaz de usuario orientada a objetos (OOUI, *Object Oriented User Interface*) y que accede a los servicios distribuidos donde quiera que se encuentren.

A fin de bajar beans y applets de Java por pedido, el cliente necesita un programa de navegación para la Web. En todo caso, el sistema operativo casi siempre pasa la responsabilidad al bloque del middleware y le permite manejar los servicios que no sean locales. El cliente también ejecuta un componente del elemento de administración del sistema distribuido (DSM, *Distributed System Management*), el cual puede ser cualquier cosa, desde un simple agente en una computadora personal hasta todo un programa de interfaz de la aplicación DSM que la convierte en una estación de administración.

El bloque básico del servidor

Ejecuta el lado servidor de la aplicación. El programa servidor suele ejecutarse sobre la parte superior de los paquetes de software para servidor. El lado servidor depende del sistema operativo para conectarse con el bloque del middleware, el cual transporta la solicitud de servicio. Asimismo, el servidor ejecuta un componente DSM, el cual puede ser cualquier cosa, desde un simple agente en una computadora personal administrada hasta todo un sistema interno o de soporte operativo de la aplicación DSM (por decir algo, podría proveer una base de datos de objetos compartidos para almacenar información de la administración del sistema).

El bloque básico del middleware

Se ejecuta en los lados cliente y servidor de la aplicación. Podemos dividir este bloque en tres categorías: pilas de transporte, sistemas operativos de red (NOS, *Network Operating System*) y middleware de servicio específico. El middleware es el sistema nervioso de la infraestructura cliente/servidor. Al igual que los otros dos bloques, el middleware también tiene un componente de software DSM.

En una red cliente/servidor; la aplicación DSM se ejecuta en todos los nodos. Una estación de trabajo de administración recopila datos de todos sus agentes en la red y los presenta de manera gráfica. Esta estación puede ordenar a los agentes que realicen acciones por ella. Visualice la administración como una actividad que se lleva a cabo en una "red dentro de otra" autónoma.

1.3.2 Capas

Con el término "capas" describiré la partición lógica de una aplicación a través de clientes y servidores. El concepto central de cliente/servidor es la división del proceso de carga. Se ha puesto en práctica la idea de dividir una aplicación en dos capas a fin de crear varios tipos de soluciones de software de red. En general, esas soluciones están integradas en la parte superior de paquetes de middleware. Sin embargo, cada una de ellas se distingue por la naturaleza del servicio que brinda a sus clientes, tipo de servidor.[*11]

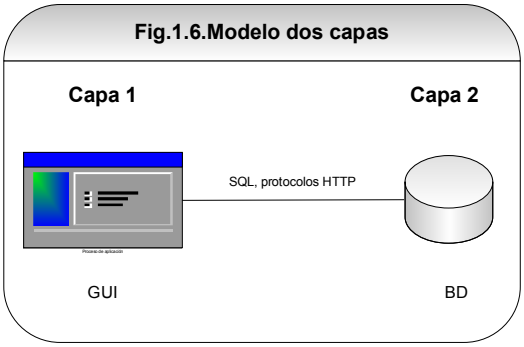
1.3.2.1 Modelo dos capas

El modelo de dos-capas divide la carga de procesamiento en dos. La mayoría de la aplicación lógica corre en el cliente, la cual típicamente envía peticiones SQL a una base de datos de un servidor residente. Esta arquitectura también es llamada fat client, porque una gran parte de la aplicación

[*11] Recopilación de información de los libros [6] y [4] p.155-157.

de lado del cliente.

En este sistema, la aplicación lógica puede estar en la interfaz de usuario sobre el cliente o dentro de la base de datos del servidor, o en ambos. Corres una GUI sobre el cliente. Este envía peticiones SQL, llamados al sistema, o los comandos HTTP sobre una red al servidor. El servidor procesa la petición y retorna los resultados. Al tener acceso a datos, los clientes tienen que conocer como es organizado y almacenado del lado del servidor. Una variación de la arquitectura 2-capas acerca del uso de procedimientos almacenados es que en lugar de enviar peticiones SQL a través de la red, los procedimientos almacenados invocan una función que corre sin una base de datos, podría pensarse como una partición de 2.5 capas.



Los servidores de bases de datos y archivos con procedimientos almacenados son ejemplos de sistemas cliente/servidor de dos capas.

1.3.2.2 Modelo 3 capas

Tres capas es un concepto que se empleó por vez primera para describir la división física de una aplicación entre computadoras personales (primera capa), servidores departamentales (segunda capa) y servidores empresariales (tercera capa). Después se utilizó para describir una división entre cliente (primera capa), base de datos local (segunda capa) y base de datos empresarial (tercera capa). Hoy en día, la definición es cliente (primera capa), servidor de aplicaciones (segunda capa) y servidor de base de datos (tercera capa).

En los sistemas cliente/servidor de tres capas, la lógica de la aplicación (o del proceso) reside en la capa intermedia, y está separada de la información y de la interfaz de usuario. Los procesos devienen cuidados de primera clase, y se les puede administrar y distribuir de manera independiente desde la GUI y la base de datos.

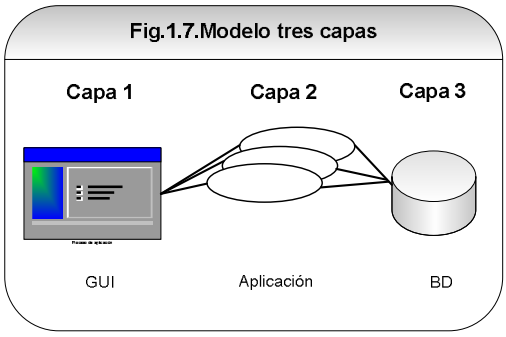
El modelo de tres capas divide la carga de procesamiento entre:

* Clientes corriendo la interfaz (GUI) lógica.
* El servidor de aplicación corriendo los negocios lógicos.
* La base de datos y/o uso de herencia.

Tres capas mueve la aplicación lógica al servidor, esto hace referencia a una arquitectura de servidor robusto o “fat server arquitectura” o más recientemente llamado “thin client”. El modelo de servidor obeso coloca más funciones (o procesos) en el servidor; el modelo de cliente obeso lo hace a la inversa.

Por definición, todas las aplicaciones cliente/servidor deben tener por lo menos dos capas: la interfaz de usuario reside en el cliente y los datos compartidos son almacenados en el servidor.

Una aplicación es de dos-capas o tres-capas basado en la separación de la aplicación lógica proveniente del GUI y de la base de datos.



Este particionamiento es una cuestión de diseño central que hace una gran diferencia en determinar el éxito de las aplicaciones de misión crítica.

Los monitores de TP, los de transacciones de objetos y los servidores de aplicaciones Web son casos de sistemas cliente/servidor de tres capas.

1.3.2.3 Comparación 2 capas & 3 capas

A continuación se presenta una comparación de los modelos de dos y tres capas de acuerdo con Robert Orfali [6]:

	2 capas	3 capas
Administración del sistema	**C**omplejo más lógica o reglas en el cliente que lo maneja	**P**oco complejo la aplicación puede ser centralmente manejada en el servidor; los programas de aplicación están más visibles para las herramientas estándar de administración del sistema
Seguridad	**B**ajo seguridad en el nivel de la información	**A**lto afinada en el nivel de servicio, método o tipo de objeto
Encapsulado de datos	**B**ajo las tablas de datos son expuestas	**A**lto El cliente invoca servicios o métodos
Funcionamiento	**P**obre muchas peticiones SQL son enviados por red; los datos seleccionados se deben descargar para su análisis en el cliente	**B**uena únicamente las peticiones y las respuestas del servicio se envían entre el cliente y el servidor
Escala	**P**obre manejo limitado de ligas de comunicaciones del cliente)	**E**xcelente concentra las sesiones entrantes; puede distribuir la carga entre varios servidores

	2 capas	3 capas
Reutilización de aplicaciones	**P**obre aplicación monolítica en el cliente	**E**xcelente puede reutilizar servicios y objetos
Facilidad de desarrollo	**A**lto	**E**n mejoría herramientas estándar pueden ser empleados para crear clientes; además, están surgiendo herramientas que pueden usarse para desarrollar los lados cliente y servidor de la aplicación.
Infraestructura servidor a servidor	**N**o	**S**í a través del middleware por el lado del servidor
Integración de aplicaciones heredadas	**N**o	**S**i via entradas encapsuladas por servicios u objetos
Soporte a Internet	**P**obre las limitaciones del ancho de banda del Internet hacen más difícil la descarga de clientes obesos (*fat clients)* y acentúan las limitaciones ya conocidas)	**E**xcelente es más fácil descargar los clientes delgados, como los applets o los beans; las invocaciones de servicio remotas distribuyen la carga de la aplicación al servidor
Soporte de Base de Datos heterogenea	**N**o	**S**i puede usar múltiples bases de datos sin la misma transacción de negocios)
Opciones de la comunicación	**N**o sólo solicitudes síncronas orientadas a conexión similares a RPC	**S**i soporta solicitudes tipo RPC, pero también mensaje sin conexión, entrega por cola, publicar y suscribir y difusión
Flexibilidad de arquitectura del hardware	**L**imitada tienes un cliente y un servidor	**E**xcelente las tres capas pueden residir en diferentes computadores, o la segunda y tercer capa pueden residir en la misma computadora, se puede distribuir la segunda capa a través de múltiples servidores en entornos basados en componentes
Disponibilidad	**P**obre	**E**xcelente puede volver a arrancar los componentes de la capa de en medio en otros servidores

1.4 SERVIDORES

Un servidor es una máquina que ejecuta su proceso a la llegada de una petición de cualquier cliente, y contiene cierto tipo de software que realiza tareas en nombre de los usuarios. Su propósito es proveer datos de modo que otras máquinas puedan utilizarlos. Cuando los usuarios se conectan a un servidor pueden acceder a programas, archivos y otra información de este.

Existe un uso dual que puede llevar a confusión. Por ejemplo, en el caso de un servidor Web, este término podría referirse a la máquina que almacena y maneja los sitios Web, y en este sentido es utilizada por las compañías que ofrecen hospedaje, mas adelante definiré este concepto. Sin embargo, el servidor Web podría referirse también al software, como el servidor de http de Apache, que funciona en la máquina y maneja la entrega de los componentes de las páginas Web como respuesta a peticiones de los navegadores de los clientes.

Los archivos para cada sitio de Internet se almacenan y se ejecutan en el servidor. Hay muchos servidores en Internet y diferentes tipos de servidores como veremos en el siguiente tema, pero comparten la función común de proporcionar acceso a los archivos y servicios. Los servidores Web, servidores de correo y servidores de bases de datos son los más comunes a los que tienen acceso la mayoría de la gente al usar Internet.

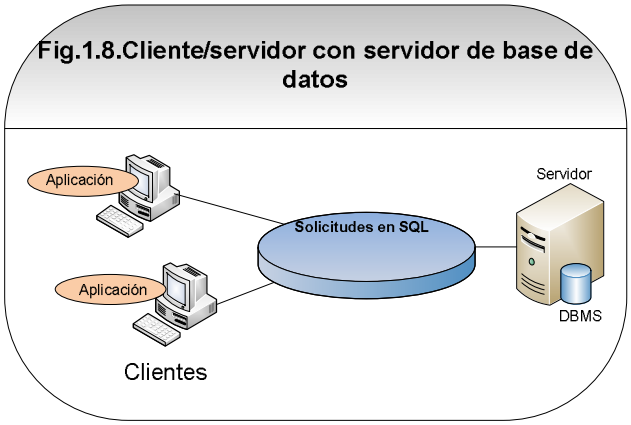
Algunos servidores manejan solamente correo o solamente archivos; mientras que otros hacen más de un trabajo, ya que una misma máquina puede tener diferentes programas de servidor funcionando al mismo tiempo, por ejemplo, un servidor que brinda servicio Web y también de Base de Datos.

1.4.1 Tipos de servidores

Existen varios tipos de servidores de acuerdo con el servicio que brindan, y cada uno de ellos funciona de distinta manera. A continuación, doy una breve descripción de los servidores más comúnmente utilizados en el ambiente laboral, de acuerdo con la bibliografía de Orfali Robert[6].

Servidores de base de datos

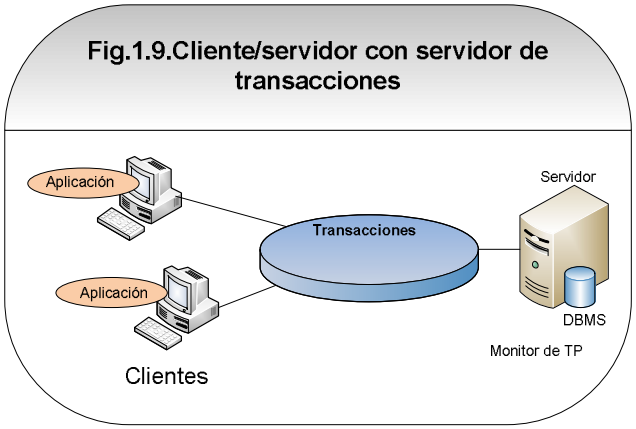
Con un servidor de base de datos, el cliente pasa como mensajes solicitudes escritas en SQL (structured query language) al servidor de base de datos **Fig.1.8.** El resultado de cada comando de SQL se devuelve por la red. El código que procesa la solicitud de SQL reside en la misma máquina, lo mismo que la información. El servidor emplea su propio poder de procesamiento para encontrar los datos pedidos en vez de entregar todos los registros al cliente y luego deja que éste encuentre lo que busca. El resultado es un uso mucho más eficaz del poder de procesamiento distribuido. En este enfoque, el fabricante del servidor del DBMS (Data Base Manager System) proporciona el código del servidor. El código de la aplicación se encuentra en el cliente, así que el código de la aplicación cliente debe estar escrito o comprar una herramienta para hacer solicitudes (o consultas) que sea ligera y compacta. Los servidores de base de datos proveen el fundamento para los sistemas de apoyo a decisiones que requieren consultas e informes flexibles. También tienen una función vital en el almacenamiento masivo de información (en almacenes de datos, datawarehousing).



Servidores de transacciones

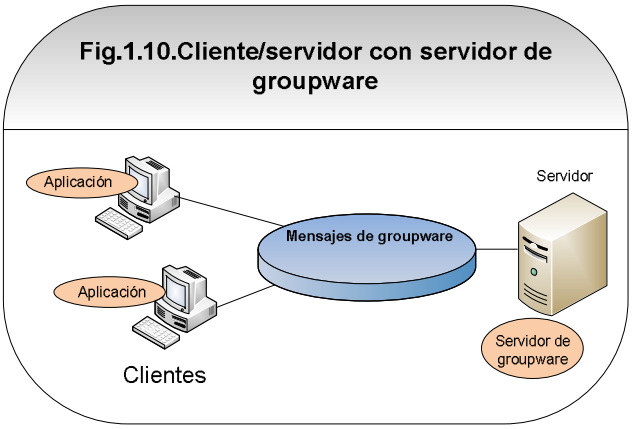
Con un servidor de transacciones, el cliente llama, mediante un motor de base de datos de SQL, a procedimientos remotos (o servicios) que residen en el servidor **Fig.1.9.** Estos procedimientos remotos (o almacenados) en el servidor ejecutan un conjunto de instrucciones de SQL. El intercambio en la red consta de un solo mensaje de solicitud/respuesta (en contraposición al enfoque del servidor de base de datos, en el que cada comando de SQL supone una transacción con un mensaje de solicitud/respuesta). En este enfoque, todas las instrucciones de SQL tienen éxito o fracasan como unidad. Tales instrucciones de SQL agrupadas se denominan transacciones.

Con un servidor de transacciones, se crea la aplicación cliente/servidor escribiendo el código tanto del componente cliente como del servidor. El componente cliente suele presentar una interfaz gráfica de usuario; el servidor consta, casi siempre, de transacciones de SQL hechas contra una base de datos. A estas aplicaciones se les conoce como de procesamiento de transacciones en línea, u OLTP (online transaction processing). En general, son aplicaciones de misión crítica que siempre requieren un tiempo de respuesta que va de uno a tres segundos. Las aplicaciones de OLTP también necesitan controles estrictos para la seguridad e integridad de la base de datos. Podemos mencionar dos tipos de OLTP: procesamiento de transacciones ligero, o TP-Lite, como se le conoce, el cual se basa en procedimientos almacenados que proporcionan los fabricantes de bases de datos; y procesamiento de transacciones pesado, o TP-Heavy, que tiene su fundamento en monitores de TP provistos por los vendedores de aplicaciones OLTP.



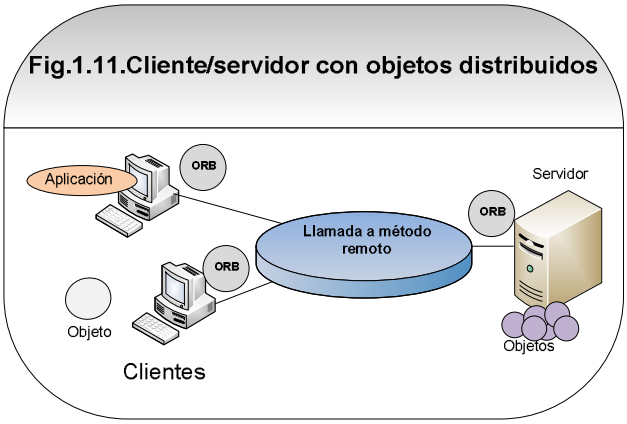
Servidores de groupware

El groupware se encarga de la administración de información semiestructurada, como texto, imágenes, correo electrónico, tableros de boletines electrónicos y flujo de trabajo. Estos sistemas cliente/servidor ponen a las personas en contacto directo con otras personas.



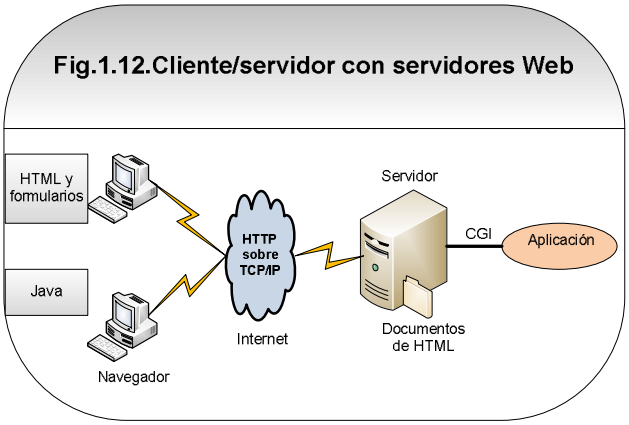
Servidores de aplicaciones de objetos

Con un servidor de objetos, la aplicación cliente/servidor está escrita como un conjunto de objetos de comunicación **Fig.1.11.** Los objetos cliente se comunican con objetos servidor mediante un intermediario de solicitud de objetos, u ORB (object request broker). El cliente invoca un método sobre un objeto remoto. El ORB localiza instancia de esa clase de objeto en el servidor de objetos, llama al método invocado y entrega el resultado al objeto del cliente. Los servidores de objetos deben dar soporte para permitir que exista concurrencia y uso compartido.



Servidores de aplicaciones Web

Un servidor Web entrega documentos cuando los clientes se los piden por nombre **Fig.1.12.** Clientes y servidores se ponen en contacto a través de un protocolo denominado HTTP (hipertext markup language), que es un protocolo que define un conjunto simple de comandos; los parámetros se pasan como cadenas de caracteres, sin disposición alguna acerca de los tipos de datos.



Un servidor Web se mantiene en espera de peticiones HTTP realizadas por un cliente HTTP, este cliente es el navegador. El navegador realiza peticiones al servidor y éste le responde con el contenido que el cliente solicita. Por ejemplo, si tecleamos un URL cualquiera en el navegador, inmediatamente éste realiza una petición HTTP al servidor correspondiente al URL. El servidor responde al cliente enviando el código HTML de la página; una vez que el cliente recibe el código, lo interpreta y lo muestra en pantalla. Con este ejemplo, se visualiza que el cliente es el encargado de interpretar el código HTML, es decir, de mostrar las fuentes, los colores y la disposición de los textos y objetos de la página; el servidor tan sólo se limita a transferir el código de la página sin llevar a cabo ninguna interpretación de la misma. Sin embargo, cuando existe una página desarrollada con el lenguaje de programación en PHP, por ejemplo, éste reside en el servidor, es decir, es interpretado por éste, de tal forma que nuestro servidor debe tener instalado, aparte del apache o servidor Web, PHP. Enseguida hago una explicación más a fondo al respecto.

Con servicio Web podemos disponer de aplicaciones Web, es decir, fragmentos de código que se ejecutan cuando se realizan ciertas peticiones o respuestas HTTP. Es interesante distinguir entre aplicaciones que se ejecutan en el lado del cliente y las que se ejecutan en el lado del servidor:

- Aplicaciones en el lado del cliente: el cliente Web, el navegador, es el encargado de ejecutarlas en la máquina del usuario. Éstas son las aplicaciones tipo Java o JavaScript; el servidor proporciona el código de las aplicaciones al cliente y éste, mediante el navegador, las ejecuta. Entonces, será necesario que el cliente cuente con un navegador con capacidad para ejecutar aplicaciones. Normalmente, los navegadores permiten ejecutar aplicaciones escritas en lenguaje JavaScript y java, aunque pueden añadirse más lenguajes mediante el uso de plugins.

- Aplicaciones en el lado del servidor: la aplicación es ejecutada en el servidor Web; entonces, genera código HTML; el servidor toma este código recién creado y lo envía al cliente por medio del protocolo HTTP.

Las aplicaciones de servidor es una buena opción para realizar aplicaciones Web aunque implique contar con mayores recursos en nuestro servidor, la razón es porque al ejecutarse en el servidor y no en la máquina del cliente, el cliente no necesita ninguna capacidad adicional; de lo contrario, si se quiere ejecutar aplicaciones JavaScript o java es necesaria la instalación de plugins y algunas herramientas extra en el navegador.

Servidores de listas

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

Un servidor de correo es una aplicación que nos permite enviar mensajes o correos de unos usuarios a otros. Para lograrlo, el servicio de correo cuenta con una serie de protocolos, cada uno con una finalidad concreta:

- SMTP, Simple Mail Transfer Protocol: Este protocolo se utiliza para que dos servidores de correo intercambien mensajes.
- POP, Post Office Protocol: Se utiliza para obtener los mensajes guardados en el servidor y pasárselos al usuario.
- IMAP, Internet Message Access Protocol: Su finalidad es la misma que la de POP, pero el funcionamiento y las funcionalidades que ofrecen son diferentes.

Así, un servidor de correo consta en realidad de dos servidores: un servidor SMTP que será el encargado de enviar y recibir mensajes, y un servidor POP/IMAP que será el que permita a los usuarios obtener sus mensajes.

1.4.2 Alojamiento de un sitio Web[*12]

Un proveedor de servicios de aplicación (ASP) ofrece alojamiento y distribución de sitios Web, soporte técnico de la aplicación, servicios de administración para los sitios Web, las operaciones, las copias de seguridad automatizadas y la seguridad, todo esto evita mucho quebradero de cabeza a las empresas. La aparición de los ASP se debe a la necesidad de cumplir con las nuevas necesidades de las empresas, independientemente de su tamaño y estructura.

Aquellos que no dispongan de tiempo, recursos financieros o personal suficiente para comprar y mantener sus propias aplicaciones de software pueden delegar en otras empresas para que lo hagan por ellos.

Los proveedores deben administrar el acceso real, la seguridad física e interna de los equipos, infraestructura como las fuentes de alimentación al sistema de aire acondicionado y todo lo concerniente a la red. Los proveedores a menudo cuentan con la tecnología más reciente y que ponen a disposición de miles de clientes. Muchas organizaciones no se pueden permitir los costos constantes que se producen por la actualización de la tecnología necesaria para mantener actualizados sus sitios Web.

Disponiendo de un buen proveedor, las cosas pueden ir bastante bien para realizar campañas de marketing; con un proveedor malo, pueden surgir numerosas dificultades.

¿Por qué es importante quién nos brinda servicios de aplicación?, existen muchos factores a considerar: la tecnología, la velocidad, capacidad de almacenamiento, disponibilidad y la cantidad de congestión en el tráfico; éstos factores son determinantes para darle rumbo y crecimiento a nuestros sitos Web.

Proveer de hospedaje Web implica hardware (servidores Web, hardware de telecomunicaciones, etc.); implica tecnología de telecomunicaciones y personal (administrador, diseñador y asistencia técnica); implica responsabilidad en cuanto a mantener la conexión durante las 24 horas del día, los siete días de la semana. Brindar un servicio fiable significa brindar más que los tres elementos que he mencionado anteriormente.

[*12] Recopilación de información de los libros [1] capítulo XXI y [4] capítulo V.

Un proveedor de alojamiento Web se encarga de administrar el almacenamiento, de supervisar el tráfico Web, de mantener el servidor Web funcionando en todo momento, se asegura de que la aplicación esté disponible a los clientes a través de Internet, independientemente de dónde se encuentren, a través de un navegador; posee y opera una aplicación de software; posee, opera y mantiene los servidores que ejecutan la aplicación. Todos los proveedores de alojamiento prometen seguridad y privacidad para sus datos, aunque no pueden garantizarlo al cien por ciento debido a que la seguridad es un tema por demás complejo.

Ventajas

- ✓ Delegar cierta parte del software a un ASP significa que la empresa puede concentrarse en sus competencias clave, en sus proyectos estratégicos, en generar ingresos y en servir a sus clientes en lugar de tener que administrar la tecnología.

- ✓ Al tener un acceso más rápido a los servicios y a las funciones más recientes, los ASP pueden mantener actualizado su entorno tecnológico mo parte de su acuerdo con el cliente. El ASP aprovecha esta ventaja distribuyendo el coste entre sus clientes.

- ✓ Los ASP también se benefician al momento de contratar personal altamente calificado para muchas empresas pequeñas que no podrían tener a tiempo completo.

- ✓ El ancho de banda de Internet se traslada al ASP, el cual puede ofrecer una velocidad muy alta a un coste bajo.

Consideraciones

- ➢ Es posible que contratar un proveedor de servicios de aplicaciones implique cierto temor en cuanto a la seguridad de los datos. Dado que el proveedor aloja el software, las empresas no pueden tener ninguna seguridad de que cualquier otra persona pueda utilizar y visualizar aquella información crítica o confidencial; esto cae en el concepto de ética profesional.

- ➢ Existe un cierto debate acerca de utilizar Internet como medio para realizar transferencias seguras de datos que son críticos. Existen numerosos virus y piratas informáticos en Internet y quizá no sea el mejor método para transferir ciertos tipos de datos. También existe el tema de la calidad del servicio que ofrece un ASP sin experiencia y su capacidad para cumplir con sus promesas.

- ➢ Con el fin de mantener un nivel de costees bajo, los ASP no suelen permitir la personalización.

- ➢ No siempre se cumple el rendimiento de la aplicación.

1.5 TIENDAS ELECTRÓNICAS[*13]

Hemos escuchado hablar de tiendas electrónicas, un concepto que otorga grandes beneficios a empresas y comercios, y también a nosotros mismos como consumidores finales. Sin embargo, éste es un tema que abarca una serie de aspectos comerciales y tecnológicos muy interesantes. Si tomamos en cuenta todo lo que entreteje éste concepto, estaríamos hablando de seguridad en primer término, de confiabilidad en los datos y cuestiones legales, por mencionar algunas. El concepto tradicional de comercio hace referencia a la actividad de intercambio de mercancías que desarrolla el comerciante.

[*13] Recopilación de información de los libros [4] p. 40-42, [5] y [1] capítulo XVII

El comercio electrónico trae consigo, a través de la simplificación de procedimientos y reducción de costes, un incremento del volumen del comercio internacional, así como de la inversión. Implica básicamente un nuevo modelo de relación empresarial basado en interacciones electrónicas que sustituyen los requisitos de presencia física de los sistemas tradicionales.

Sin embargo, una definición de comercio electrónico que nos da Oscar R. González[5] es "la realización de la actividad de intercambio a través de un medio electrónico." Específicamente, implica la realización de la actividad comercial de intercambio asistida por las telecomunicaciones y herramientas basadas en ellas. Esto supone un nuevo enfoque a la hora de entender la relación de intercambio entre comprador y vendedor, ya que en este caso las partes interactúan electrónicamente.

Muchas tecnologías pueden ser usadas como apoyo del comercio electrónico, resaltando el hecho de que es algo más que simplemente un fenómeno basado en Internet. El comercio electrónico abarca tecnologías de comunicaciones como el intercambio electrónico de datos (EDI), la transferencia electrónica de fondos (EFT), soportes multimedia, aplicaciones relacionas con las rede de comunicación: correo electrónico, tablones electrónicos de anuncios, transferencia de archivos (FTP); videoconferencia, etc.

Ventajas del comercio electrónico

Esta nueva forma de realizar transacciones u operaciones comerciales es, para las empresas, una forma alternativa o complementaria de realizar sus actividades y, para los clientes, un nuevo entorno a la hora de afrontar el proceso de compra, presentando una serie de ventajas para ambas partes.

- ✓ Permite hacer más eficientes las actividades de cada empresa.
- ✓ Aumento de las ventas y la competitividad
- ✓ Establece formas más dinámicas de cooperación entre empresas.
- ✓ Reduce las barreras de acceso a los mercados actuales, en especial para pequeñas empresas.
- ✓ Facilita un acceso más directo y sencillo a la empresa.
- ✓ Para el consumidor, amplía su capacidad de acceder a prácticamente cualquier producto y de comparar ofertas, permitiéndole además convertirse en proveedor de información.
- ✓ Reduce o incluso elimina por completo los intermediarios, por ejemplo, en la venta de productos en soporte electrónico (textos, imágenes, videos, música, software, etc.), que se pagan y entregan directamente a través de la Red.
- ✓ Ofrecer una imagen empresarial de vanguardia.
- ✓ Trabajar a escala mundial sin establecer oficinas en países extranjeros

Consideraciones

A pesar de sus múltiples ventajas, plantea también problemas nuevos o agudiza algunos de los ya existentes en el comercio tradicional, entre ellos:

- La validez legal de las transacciones y contratos "sin papel".
- La necesidad de acuerdos internacionales que armonicen las legislaciones sobre comercio.
- El control de las transacciones internacionales, incluido el cobro de impuestos.
- La protección de los derechos de propiedad intelectual.
- La protección de los consumidores en cuanto a publicidad engañosa o no deseada, fraude, contenidos ilegales y un uso abusivo de los datos personales.
- La seguridad de las transacciones y medios de pago electrónicos
- Dificultad de localizar las tiendas debido a la gran cantidad de sitios existentes y a la inexperiencia en el uso de la Red.

Como se ha venido mencionando a lo largo de este tema, uno de los aspectos más importantes en el ámbito del comercio electrónico es la seguridad. El desarrollo del comercio electrónico requiere la

existencia de un entorno legal básico en aspectos como éste, los derechos de propiedad intelectual, los impuestos, etc. Los avances técnicos obligan a revisar sobre la marcha las leyes actuales, con el riesgo de que diferentes países adopten criterios incompatibles. Como ejemplo los sistemas de cifrado, en unos países pueden ser más estrictos que en otros.

La apertura de una tienda electrónica en definitiva es más complicada que realizar una página con la cual simplemente se pretenda tener presencia en Internet. En primer lugar, se necesitan más trámites administrativos. En segundo lugar, para construir una tienda virtual de calidad es necesario contar con un servidor seguro con tecnología SSL. De esta forma, los clientes podrán enviar los datos de su tarjeta de crédito o de otro método de pago on-line, con la confianza de que se hace de forma segura en un formato encriptado que evite que otros vean esa información.

1.5.1 Componentes del comercio electrónico[*14]

Es importante contar con el sistema (software), es decir, una tienda electrónica y el hardware que lo alberga y soporta. A continuación, menciono los requerimientos y los elementos que conforman el sistema:

El cliente

A pesar de todas las ventajas positivas y prometedoras que ofrece el comercio electrónico, sigue sufriendo los inconvenientes derivados de su propia naturaleza: la automatización que suprime el contacto humano entre comprador y vendedor, y que a ojos del consumidor, es considerado una falta de atención. Todo lo que se pueda hacer por mejorar la relación entre ambos servirá para construir sólidos puentes de efecto duradero, que al final, es el apoyo al cliente y el servicio prestado lo que genera los auténticos dividendos. El cliente es la parte fundamental para nuestro sistema, por esta razón algunos de los servicios más importantes que debe presentar la tienda electrónica al cliente de acuerdo con Linda G. Lara Vivas[12]son:

1. Registro como cliente.
2. Actualización de sus datos personales.

[*14] Información basada en el artículo enter@te, divulgación sobre cómputo, Internet y telecomunicaciones [12] y en el libro [4] capítulo XV.

3.Identificación a través de su correo electrónico y contraseña.
4.Recuperación de su contraseña en caso de extravío.
5.Registro de sus preferencias.
6.Consulta de productos con base en sus preferencias.
7.Consulta de la historia de sus pedidos y compras realizadas.
8.Envío de correos electrónicos al momento de su registro durante las diferentes partes del proceso de su compra (confirmación del pedido, recepción del pago, realización del envío).

De forma adicional, el cliente puede obtener información de:

1.Ayuda.
2.Políticas de envío.
3.Políticas de devoluciones.

Catálogo de productos

Presenta la información de los productos que debe ver el cliente. Es recomendable que los productos o servicios sean presentados de la mejor manera posible para facilitar la búsqueda en la base de datos.

Esta función debe permitir a los usuarios buscar aquello que necesitan.

Carrito de compras

Funciona como un carrito de compras de verdad, permitiendo a los clientes ir recogiendo artículos y guardarlos ahí hasta el momento de la compra final, pueden añadir o descartar productos mientras miran las opciones que les ofrece el catálogo o la base de datos. Éste mecanismo permite al cliente mantener un registro de los artículos que desea adquirir, manteniendo el control del número de artículos y el precio asignado.

Sistema de procesamiento de pedidos

Tramita las órdenes de compra: suma los totales, calcula los impuestos, los gastos de transporte y la información de envío (la dirección y datos para generar su factura); determina el método de pago de los artículos (depósito bancario o tarjeta de crédito), y genera informes detallados de ventas y de clientes. Finalmente, el cliente recibe una confirmación de su pedido en pantalla.

Recepción de pagos

En el mundo real, existen tres formas de pagar los bienes: efectivo, cheque o tarjeta de crédito. Las tarjetas pueden ser inteligentes, de débito, cajero automático y cualquier otra clase de tarjeta de crédito, ya que todas sirven para permitr a los consumidores pagar sin tener que utilizar dinero en efectivo. Son los llamados medios de pago electrónico en línea.

Todo entorno de comercio electrónico con un sistema de pago rquiere un diseño complejo que garantice la seguridad del pago, la privacidad de la transacción, la integridad del sistema la utentificación del cliente y la promesa del comprador de que va a pagar, dicha aplicación debe contar con sistemas de seguridad muy estrictos.

Entregar el producto

Una vez que un comprador ha elegido lo que quiere y lo ha pagado, llega el momento de que el comerciante entregue los bienes con la mayor prontitud. La velocidad de la entrega es un factor clave, sobre todo en el caso de productos que se descargan desde Internet, como música o paquetes de software, donde los compradores esperan que la entrega sea inmediata. Si los artículos son físicos como prendas o libros, los compradores esperan que les lleguen como mínimo tan rápido como si hicieran el pedido por teléfono. Resolver los problemas de transporte es un factor decisivo que puede determinar el éxito o el fracaso de un negocio electrónico.

El comercio electrónico mejora la forma de efectuar actividades comerciales, es una de las principales puertas de entrada a la sociedad de la información, aprovecha las ventajas de comunicación y acceso a la información que ofrece "la Red de Redes". Sin embargo, aún falta mejorar detalles en cuanto a la seguridad de la información.

CAPÍTULO II ALTA DISPONIBILIDAD Y BALANCEO DE CARGA

Hoy en día, las organizaciones cuentan con sistemas de cómputo que ofrecen servicios o aplicaciones que por su naturaleza deben proporcionar un servicio ininterrumpido de 24 horas al día, 7 días a la semana. Los sistemas operativos Unix/Linux son conocidos como sistemas operativos estables; sin embargo, existen otros factores que conllevan a sufrir alguna falla, como la fiabilidad del hardware o algún fallo humano debido a un error en la administración del sistema.

Los equipos de cómputo siempre están susceptibles a sufrir alguna falla, sin embargo, hay que diferenciar el grado en que afecta a los usuarios, la producción, el desarrollo, etc. Dependiendo de la naturaleza del servicio y de los recursos con los que se cuentan es la solución que se debe proporcionar.

El proyecto de High Availability Linux (*Alta Disponibilidad*) pretende brindar una solución de alta disponibilidad para Linux que promueve la confiabilidad, disponibilidad y capacidad de servicio a través de un esfuerzo de desarrollo de la comunidad de programadores que ayudan al proyecto.

En este capítulo pretendo introducir al lector en el tema de alta disponibilidad, definir conceptos relacionados con la misma y profundizar en el funcionamiento del concepto de UltraMonkey y sus topologías.

2.1 DISPONIBILIDAD EN LOS SISTEMAS[*15]

La disponibilidad en los sistemas debe ser brindada para aplicaciones de vital importancia, desde el punto de vista de la empresa, por ejemplo, servicios que le brindan ganancias, quizá también el correo electrónico por su importancia dentro de la empresa, etc. La práctica de ésta, ayuda a reducir el tiempo de inactividad en el entorno de la aplicación, además, el uso de buenas prácticas de la tecnología de la información y de topologías de hardware puede mejorar en gran medida la disponibilidad.

El tiempo de inactividad de la aplicación o aplicaciones perjudica en gran medida a las empresas que pueden ver la disminución de ventas, productividad y pérdida de confianza por parte de los clientes.

La implementación de soluciones de alta disponibilidad mejora en mucho la presencia de la aplicación o servicio en Internet o la Intranet, reduciendo de esta forma el tiempo de inactividad previsto como tareas de mantenimiento o instalaciones; y como el imprevisto, un error del servidor, errores humanos, fallas de suministro eléctrico, etc.

2.1.1 Métrica de disponibilidad

La forma en que se mide la disponibilidad es mediante el porcentaje de tiempo que el sistema es capaz de realizar las funciones para las que está diseñado sin tiempos fuera.

La disponibilidad suele medirse en “nueves”. Para muchas aplicaciones el 99 por ciento de la disponibilidad es adecuada; una solución cuyo nivel de disponibilidad sea de “tres nueves” es capaz de realizar su función prevista el 99.9 por ciento del tiempo, lo que equivale a un tiempo de inactividad anual de 8 horas 45 minutos por año sobre una base de 24 horas al día, siete días a la semana los 365 días al año.

[*15] Recopilación de información de [17] y [21], e-book[11]

En la **Tabla 2.1.** se muestran los niveles de disponibilidad frecuentes que muchas organizaciones intentan conseguir:

Porcentaje de disponibilidad	Inactividad para días de 24 horas	Inactividad para días de 8 horas
90%	876 horas (36.5 días)	291.2 horas (12.13 días)
95%	438 horas (18.25 días)	145.6 horas (6.07 días)
99%	87.6 horas (3.65 días)	29.12 horas (1.21 días)
99.9%	8 horas 45 minutos	2 horas 54 minutos
99.99%	52 minutos 33 segundos	17 minutos 28 segundos
99.999% [*16]	5 minutos 15 segundos	1 minutos 44 segundos
99.9999%	31 segundos	10 segundos

Tabla 2.1. Niveles de disponibilidad

Medir o elegir el porcentaje de disponibilidad nos es tan sencillo como seleccionar uno de los porcentajes que se muestran en la tabla anterior, ello implica decidir qué factores medirán la disponibilidad, es decir, la métrica que medirá el tiempo de inactividad. Por ejemplo, una organización puede considerar que se produce tiempo de inactividad cuando una base de datos no está montada, otra organización puede considerar que sólo se produce tiempo de inactividad cuando más de la mitad de sus usuarios se ven afectados por una interrupción del servicio.

Además, los requisitos de disponibilidad deben establecerse en el contexto del servicio y de la organización que utiliza el servicio. Por ejemplo, los requisitos de disponibilidad para los servidores que alojan datos de carpetas públicas que no son críticos pueden ser inferiores que los de los servidores que contienen bases de datos con información fundamental para la empresa.

2.1.2 Niveles de disponibilidad

Los niveles de disponibilidad que a continuación se mencionan son arbitrarios pero representan escenarios reales que van desde el nivel más básico hasta el más completo. Entre cada uno de los niveles de disponibilidad existe la posibilidad de variantes que incrementen o decrementen la disponibilidad del sistema

Nivel 1. Disponibilidad regular:

Es el nivel básico de protección del sistema, en éste, no contamos con medidas de redundancia especiales. En este nivel, solo contamos con los respaldos de datos que se deben hacer periódicamente, no existen planes de contingencia en caso de que el sistema falle. Si por ejemplo, se tiene una falla de la máquina que no nos permita recuperar la información de nuestro disco, dependiendo de la periodicidad con la que se realizan los respaldos, es la información que recuperaremos. Este nivel de disponibilidad implica bastantes consecuencias, entre ellas: la no disponibilidad del servicio, que dependiendo de su naturaleza, son las implicaciones de costes a la que la empresa se enfrentará; la recuperación de datos no actualizados debido a la recuperación únicamente por los respaldos realizados. Este nivel muy probablemente no satisfaga las necesidades de las empresas con grandes masas de datos y transacciones.

Nivel 2. Disponibilidad incrementada:

El nivel 2 tiene la diferencia con el anterior nivel, en algunas medidas de protección de datos,

[*16] Los también llamados "cinco nueves", es un porcentaje alto, difícil de alcanzar pero aún es posible lograrlo.

esto significa emplear tecnología RAID y entonces la falla de un disco no será pérdida de datos debido a que la información es almacenada en más de un disco físico, tal vez haya la inactividad del sistema pero con el refuerzo de los respaldos el más crítico recurso del sistema que son los datos estarán protegidos por algún esquema implementado de RAID en los discos duros.

Nivel 3. Alta disponibilidad:

Este nivel es normalmente llamado High Availability y en esta configuración se tomarán dos servidores o más que formarán un conjunto actuando como un solo sistema (cluster), se puede combinar con discos RAID; el sistema tendrá componentes duplicados, si uno falla, de forma automática o manual serán activados para reemplazar el que esté fallando.

Aún en estos sistemas existe tiempo fuera debido a la inactividad, pero en muchos casos este tiempo es limitado, en este nivel puede ser alcanzada la disponibilidad de un 99.98% e incrementarlo implementando protecciones adicionales en otros puntos de falla o SPOF. El implementar HA requiere de un costo considerable debido a que cada componente tendrá su redundante lo cual resultará tener dos sistemas y sólo uno de ellos será el activo, mientras el otro estará a la espera de la señal de una falla.

Nivel 4. Recuperación en desastres

Este es el nivel de disponibilidad más alto y caro de protección de un sistema, cuando este esquema es implementado se estará protegiendo en contra de la pérdida total del sitio de operaciones teniendo un sitio alterno a distancia del original y éste necesitará tener todo lo necesario para operar de igual forma en caso de desastre.

2.1.3 Medición de la disponibilidad

De forma general, la disponibilidad se puede calcular con la siguiente fórmula:

$$D = (HDA - HF)*100/HDA$$

Donde:

D = disponibilidad
HDA = horas que el servicio debería estar activo al año
HF = horas totales anuales que el sistema se encontró fuera de servicio.

Si las horas totales que el sistema se encontró fuera de servicio aumenta, nuestra disponibilidad anual disminuye. Si reducimos las horas totales anuales que el sistema debe estar activo, nuestro porcentaje de disponibilidad mejora considerablemente.

Por ejemplo, el tiempo fuera anual de un dominio cualquiera es de aproximadamente una hora, ¿cuál es la disponibilidad con la que cuenta el sistema?

$$D=(8760-1)*100/8760$$
$$D=99.988\%$$

Como ya había comentado, deben definirse bien los parámetros mediante los cuales nos vamos a basar para hacer el cálculo de la disponibilidad, éstos se refieren también al tiempo que el servicio tenga que estar en función.

2.1.4 Costes intangibles

Por muy poca importancia que le queramos dar a éste tema, hay que tomarlo en consideración porque pone de manifiesto algunas cosas en las que hay que pensar considerablemente.

Por ejemplo, la pérdida de lealtad por parte de los clientes, es decir, después de cualquier incidencia que impida a la empresa satisfacer a su cliente durante una baja del sistema, es un buen pretexto para que el usuario busque en otro lugar. Quizá un cliente no sea mucho, pero si tomamos en cuenta tiempos fuera de nuestro sistema de forma consecutiva, seguramente la pérdida de clientes sea considerable.

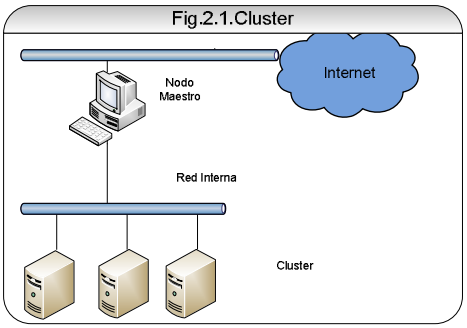
Quizá por mencionar, podamos considerar un efecto secundario sobre clientes de los clientes insatisfechos.

Evaluar los efectos de los tiempos fuera del sistema no se pueden medir con facilidad, sin embargo, es necesario tomarlo en cuenta para saber a cuánto ascienden los efectos y cuáles son las consecuencias de permitir bajas en nuestro sistema. Recordemos que el alcance de los efectos de un fallo, no depende solamente de su duración; su cálculo supone también la evaluación de si durante todo el tiempo del fallo del sistema, los datos realmente necesitaban ser disponibles.

2.2. CLUSTER[*17]

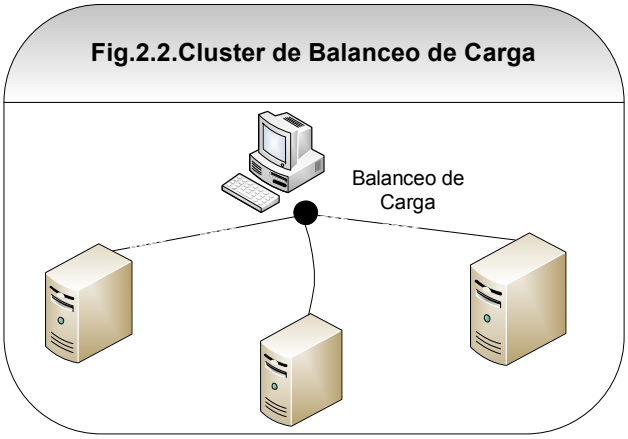
Un cluster es un tipo de paralelismo o sistema de procesamiento distribuido, el cual consiste de una colección de equipos independientes que se encuentran trabajando en conjunto como si fueran uno solo, integrando recursos de cómputo. Un cluster generalmente se refiere a dos o más computadoras conectadas entre sí. Las máquinas o también llamadas nodos pueden existir en un gabinete o estar físicamente separadas y conectadas vía una LAN (*Local Area Network*). La interconexión de un cluster de computadoras pueden aparecer ante el usuario como si fuera un solo sistema.

La implementación de clusters permiten aumentar la disponibilidad, fiabilidad y escalabilidad de múltiples niveles de red. Desde un punto de vista general, un clúster**,** debe contar con el software necesario: un sistema operativo confeccionado especialmente para esta tarea, compiladores y aplicaciones especiales que permite que los programas que se ejecutan en el sistema exploten todas las ventajas del cluster; y con la interconexión hardware entre las máquinas del cluster. Se han desarrollado interfaces de interconexión especiales muy eficientes, pero es común realizar las interconexiones mediante una red Ethernet dedicada de alta velocidad. Gracias a esta red de interconexión los nodos del cluster intercambian entre sí las tareas, las actualizaciones de estado y los datos del programa. En un cluster abierto, existirá una interfaz de red que conecte al cluster con el mundo exterior (Internet) **Fig.2.1.**



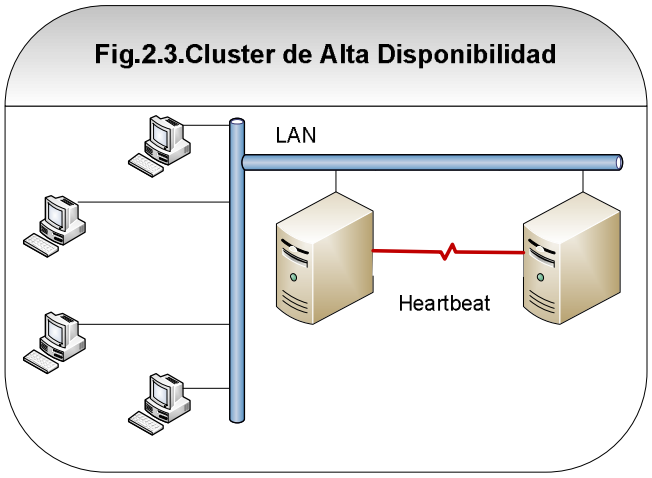
[*17] Información relevante acerca de cluster, recopilación de información del libro [8] capítulo I p. 9 y 10 y referencia [21]

En el caso anterior, las máquinas o nodos del clúster representan una máquina, sin embargo, la situación es totalmente distinta para los clusters de Balanceo de Carga **Fig.2.2.**, en éste tipo de configuración existe una máquina director y el cluster de máquinas que contienen la aplicación, el hardware y el software deben actuar en conjunto para que el tráfico se distribuya entre los nodos del cluster. Es decir, las máquinas que contienen la aplicación son idénticas, de esta forma se pueden ofrecer los servicios a mayor velocidad o se realiza una tarea de manera más rápida.



La función de los servidores de un cluster de alta disponibilidad **Fig.2.3.,** es la de estar preparados para entrar inmediatamente en funcionamiento en caso de que falle algún otro servidor.

Un cluster de alta disponibilidad es un conjunto de dos o más máquinas que contienen la misma aplicación y porque están constantemente monitoreándose entre sí. En caso de producirse un fallo del hardware o de las aplicaciones de alguna de las máquinas del cluster, el software de alta disponibilidad es capaz de rearrancar automáticamente los servicios que han fallado en cualquiera de las otras máquinas del cluster de alta disponibilidad. Y cuando la máquina que ha fallado se recupera, los servicios son nuevamente migrados a la máquina original. Esta capacidad de recuperación automática de servicios nos garantiza la integridad de la información, ya que no hay pérdida de datos, y además evita molestias a los usuarios, que no tienen por qué notar que se ha producido un problema. De tal forma que el cliente no notará ningún tiempo fuera de servicio.



Existen una variedad de paquetes software para la instalación y configuración de un cluster de alta disponibilidad bajo Linux.

Construir un cluster puede aportar bastantes ventajas en una variedad de aplicaciones y ambientes:

- Brinda mayor velocidad de respuesta debido a los clusters de Balanceo de Carga de tal forma que influye en el incremento del número de transacciones.
- Por ende, los clusters de alta disponibilidad incrementan la confiabilidad y robustez del sistema

Es necesario diferenciar entre un cluster de alto rendimiento y los clusters de alta disponibilidad, el cluster de alto rendimiento implica una configuración de equipos diseñado para proporcionar capacidades de cálculo mucho mayor que las que proporcionan los equipos individuales (por ejemplo los sistemas de tipo Beouwulf), sin embargo, los clusters de alta disponibilidad están diseñados para garantizar el funcionamiento ininterrumpido de ciertas aplicaciones.

Los cluster dan lugar a la escalabilidad, es decir, permiten aumentar la capacidad de un equipo de tal forma que pueda crecer en cuanto a recursos para hacer frente a volúmenes de trabajo cada vez mayores sin olvidar el rendimiento. La escalabilidad nos permite crecer verticalmente, esto es en cuanto a hardware, se basa en la utilización de un gran equipo cuya capacidad se aumenta a medida que lo exige la carga de trabajo existente.

Nos brinda disponibilidad, es decir, la oportunidad de estar presente, listo para usarse. Nos brinda fiabilidad, alta probabilidad en un funcionamiento correcto. Aunque hasta el momento hemos aprendido que hasta el más confiable de los sistemas puede llegar a fallar.

2.3 SERVICIO DE DATOS[*18]

Se entiende como servicio de datos a la aplicación en sí, qué información ofrece y el servicio que brinda a sus clientes. Qué tan importante es el servicio que brinda, dado que puede ser interrumpido, qué medidas tomar para disponer de disponibilidad en el servicio. Una vez decidido brindar disponibilidad al servicio, debe ser viable la flexibilidad entre los nodos del clúster, de forma que en caso de fallo de alguno de los nodos, éste pueda ser suplantado físicamente por los nodos restantes, evitando que el servicio de datos que se brinda cambie. Esta "virtualización" del recurso permite que una máquina pueda suplantar la función que desempeña otra dentro del cluster. El servicio de datos únicamente se verá afectado con el tiempo de cambio necesario para el conjunto de recursos y la puesta en marcha del servicio de datos.

Un ejemplo sería el servicio de correo electrónico que brinda una empresa, es un servicio prioritario. Para brindarlo, se necesita una máquina dónde será ejecutado, un sistema de archivos y directorios dónde guardar los correos recibidos, contactos, etc., y de una red dónde poder responder a las peticiones del servicio. Para alta disponibilidad, éstos recursos deben atender a cierta flexibilidad, es decir, tener la capacidad de seguir brindando el servicio (recurso estático virtualmente) y recurso dinámico físicamente (que otra máquina tome la función del nodo que esté fallando).

2.3.1. Recursos computacionales

Como recursos computacionales son considerados los nodos del cluster y las máquinas, es decir, recursos físicos que permiten que el programa que se encarga de ofrecer servicio de datos pueda ser ejecutado. Físicamente, nuestras máquinas o nodos deben ser idénticas, es decir, cada una de ellas debe tener instalado el programa del servicio de datos para que al momento de ofrecer el servicio no muestre diferencia alguna.

[*18] La información referente a los recursos computacional, de comunicaciones y almacenamiento fue apoyada en la referencia [22]

Para el concepto de alta disponibilidad, se debe definir la máquina que será el maestro del servicio dependiendo del estado del clúster. Y por medio del software se decide qué nodo va a alojar qué servicio de datos.

2.3.2 Recursos de comunicaciones

Al servicio de datos se accede mediante una red de comunicaciones. Esto implica las interfaces de red así como el conjunto de protocolos, los cuales tienen como función responder a varias direcciones de IP con el fin de dar flexibilidad al servicio de datos; la flexibilidad tiene mucho que ver con la virtualización del servicio, tema LVS (*Linux Virtual Server*). Para el protocolo TCP/IP el servicio de datos será accedido mediante una dirección IP y un puerto: para que el servicio de datos pueda residir físicamente en cualquier nodo se deben utilizar IP's virtuales para que esto sea posible.

Si se utiliza una red Ethernet-TCP/IP y el NIC (*Network Interface Card*) no es capaz de cambiar la dirección MAC (*Media Acces Controler*), se puede llegar a un problema con las tablas ARP de los demás elementos de red; ya que se debe obligar a los clientes o routers de la LAN a actualizar la nueva dirección MAC para la IP del servicio. Más adelante se analiza este problema.

2.3.3 Recursos de almacenamiento

Los recursos de almacenamiento es un punto delicado por tratar debido a que es en éstos recursos donde se deposita la información y la mayoría de los casos es información valiosa y con la que se debe tener especial cuidado. En estos recursos de almacenamiento tendremos la aplicación que se usará para el servicio de datos junto con los datos, es decir, la aplicación y los datos colocados en una base de datos.

El almacenamiento se desarrolla de una forma complicada para configuraciones clásicas debido que la información debe ser manejada con especial cuidado. En configuraciones hardware más modernas no hay tanta problemática, la solución de las arquitecturas SAN (*Storage Area Network*) permiten que los recursos de computación accedan por red SAN a los recursos de almacenamiento. Sin embargo, si no se cuenta con los recursos o la naturaleza de la aplicación no requiere de una SAN para implementar este tipo de configuraciones, los recursos de almacenamiento suelen ser servidores con arreglo de discos RAID o un conjunto de máquinas que configuran un cluster. El entorno SAN permite acceder a los recursos de almacenamiento de forma muy flexible.

El recurso de almacenamiento debe contar con flexibilidad para la alta disponibilidad, debe ser independiente para las necesidades de cada servicio de datos.

Se le llama grupo de volúmenes al conjunto de volúmenes que pertenecen a un servicio de datos en concreto. Cada volumen es un espacio de almacenamiento que el servicio de datos utilizará para sus propósitos con independencia de los demás. Es de vital importancia que el recurso de almacenamiento sea capaz de mantener la integridad de los datos y el tiempo de recuperación ante cualquier fallo no cause mayor problema.

Ante problemas de fallo existen técnicas de journaling para la administración de los datos. Journaling es una técnica que nació de las bases de datos y se ha ido incorporando a los sistemas de archivos. Cuando un sistema de archivos sufre una caída debido a un fallo del sistema, este se chequea al completo corrigiendo las inconsistencias.

2.4 DINÁMICA de Alta Disponibilidad

Entendamos como dinámica de alta disponibilidad todo lo concerniente a las configuraciones que deban realizarse en el cluster de tal forma que garanticen la máxima disponibilidad del servicio de datos. Esta actividad se desarrolla en los nodos integrantes del cluster. Todos aquellos problemas que puedan surgir y provoquen baja temporal del servicio se definen a continuación:

2.4.1 SPOF (Single Point Of Failure)

Punto simple de fallo

Un SPOF puede ser hardware, algún componente eléctrico, software o cualquier elemento que pueda estar sujeto a fallos, afectando con ello el servicio. Para brindar a un sistema alta disponibilidad la regla básica consiste en la replicación de los elementos. Para brindar alta disponibilidad se deben identificar los SPOF y evitarlos, ya que en caso de fallo puede peligrar el servicio de datos.

Si en un entorno de servidores falla algún elemento que impida proveer el servicio y no puede ser reemplazado fácilmente por otro, en los tiempos determinados, el servidor se considerará como SPOF.

Si partimos de los conceptos anteriores de alta disponibilidad, la configuración de cluster de máquinas, con hardware y software adecuado podemos reemplazar una máquina completa de forma automática. Es decir, si los puntos de falla se encuentran en los discos duros o en la tarjeta de red y alguno de estos falla, entonces levantará automáticamente la otra máquina y brindará servicio mientras el problema es resuelto.

Hay que tomar en cuenta hasta que nivel se puede brindar alta disponibilidad, pues habrá un nivel en que la alta disponibilidad se haga extremadamente cara, por ejemplo un caso extremo, consideremos que sólo tenemos un único edificio para nuestro sistema, perfectamente podemos considerarlo como SPOF en caso de incendio en el edificio, sin embargo, éste es hasta cierto punto extremo, pues podrían existir otras cosas más importantes que el servicio. Ahora, en caso de considerarlo, se puede colocar una de las máquinas del cluster en otra sede.

2.4.2 Failover

Failover se asegura de que en caso de que el servidor primario esté deshabilitado por cualquier razón, los usuarios automáticamente seguirán trabajando con el servidor secundario, es decir, el servidor secundario asume la responsabilidad del primario, importa sus recursos y levanta el servicio de datos. Una situación de failover es una situación especial para la cual la alta disponibilidad ha sido desarrollada, el fallo de una máquina o nodo. Dependiendo de las máquinas con las que se haya conformado el cluster de alta disponibilidad es el SPOF en que se encontrará el sistema, es decir, si contamos con solo dos máquinas en nuestro cluster, tras el fallo de uno de ellos el otro debe tomar su lugar (failover), sin embargo, al tener abajo una de las máquinas, el sistema se encuentra en una situación de SPOF hasta que el administrador del sistema verifique y restaure el cluster. El failover nos permite tener disponible el servicio, que es el objetivo de la alta disponibilidad.

2.4.3 Takeover

El takeover es un failover automático, igual que el failover, se produce cuando existe un fallo en el servicio de datos, sin embargo, en este caso, el procedimiento es automático. Para que este procedimiento suceda debe existir un monitoreo con respecto al servicio de datos y a la máquina.

El servidor que se encuentra fallido es forzado a ceder el servicio y recursos.

2.4.4 Switchover o Giveaway[*19]

El switchover o giveaway es un failover manual, consiste en ceder los recursos de un servicio de datos y los datos, a otro nodo del clúster, mientras se realizan ciertas tareas administrativas. A este procedimiento se le denomina "Node outage".

2.4.5 Splitbrain[*19]

Por la forma como funciona un cluster de alta disponibilidad es necesario un mecanismo de comunicación y verificación entre nodos integrantes. Por medio de este mecanismo, cada máquina o nodo debe administrar los recursos que corresponden a cada uno, según el estado del cluster; a su vez, cada nodo debe hacer chequeos a las demás máquinas por medio de 'beats' (latidos).

Un Splitbrain es un caso especial de failover, en este caso, si falla la administración del cluster de dos máquinas debido al mecanismo de comunicación se desata una situación en la cual cada máquina del cluster cree que debe tener el control debido a que para él, la máquina compañera está inactiva puesto que no puede saber su estado. Entonces tomará acciones en consecuencia, forzando un takeover.

Esta situación es un tanto delicada, puesto que los dos nodos intentarán apropiarse de todos los recursos, incluyendo el servicio de datos. El problema se agudiza cuando tenemos recursos delicados, como recursos de almacenamiento, ya que cada nodo podría tomar y escribir por su cuenta, y quebrar de esta forma la integridad de los datos.

Para evitar este problema se pueden tomar dos medidas: la primera, que cada nodo actúe de forma prudente, y utilizar los recursos compartidos como señal de que se está vivo. La forma de proceder de cada nodo es similar, ya que cada uno cree que su compañero ha desaparecido. Después de que un nodo aprecia este problema, tiene indicado que reserve un recurso llamado quorum. Un recurso quorum, es un recurso compartido, que se ha preestablecido en ambos nodos como tal. Este recurso es exclusivo, sólo un nodo del cluster puede reservarlo. Como este recurso sólo puede ser reservado por un nodo, el nodo que llegue tarde a la reserva del recurso, entiende que debe abandonar el cluster y ceder todos sus recursos. El quorum es utilizado como simplemente, método de decisión. La segunda forma de evitar esta situación es que un nodo elimine a su compañero, esta forma es un tanto violenta debido a que la ley a seguir es "el primero que apague a su compañero se queda con todos los recursos". Es un mecanismo muy brusco, pero muy eficaz.

2.5 SOLUCIÓN DE ALTA DISPONIBILIDAD PARA LINUX

En el desarrollo de código abierto han surgido proyectos que dan solución a la alta disponibilidad en ambientes Linux, de los cuales algunos destacan por su estructura un tanto compleja y en otras la sencillez que brinda en cuanto a configuración se refiere. Entre éstas soluciones se encuentra el proyecto UltraMonkey, el cual, como veremos en este tema, proporciona distintas topologías que dan solución a diferentes problemáticas. UltraMonkey es uno de los proyectos funcionales que más se ha usado y que integra herramientas de software libre.

2.5.1 ULTRAMONKEY [19]

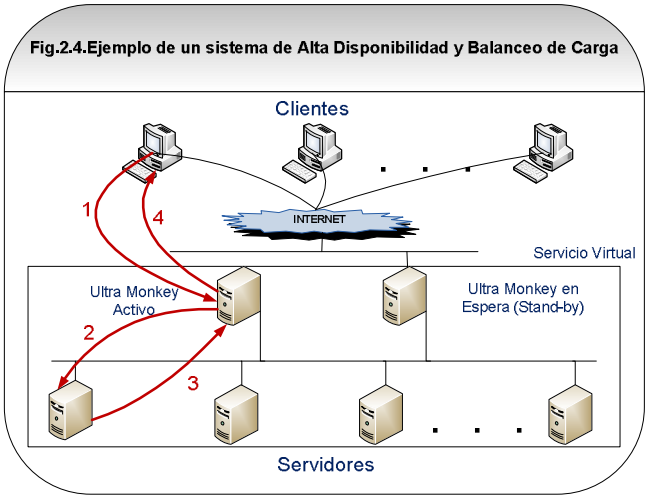
Actualmente existen muchos proyectos destinados a proveer de alta disponibilidad a un sistema, uno de ellos es UltraMonkey. UltraMonkey es un proyecto que brinda balanceo de carga y

[*19] Juan P. Paredes explica conceptos básicos de alta disponibilidad, disponible en Internet vía Web. URL[23]

alta disponibilidad de servicio, integra distintas herramientas de software libre en redes de área local. Más adelante explicaré a detalle sus herramientas de software libre, LVS y Heartbeat.

Un ejemplo de lo que nos puede brindar el proyecto de UltraMonkey lo podemos visualizar en la **Fig.2.4.**

El objetivo de mostrar UltraMonkey, es conocer los elementos que lo conforman, entender la funcionalidad de cada uno de ellos para que al final pueda haber una compresión completa de las topologías.



Funcionamiento:

1. Un usuario final envía un paquete al servicio virtual
2. UltraMonkey selecciona un servidor y reenvía el paquete
3. El servidor una vez procesada la información la envía de regreso
4. UltraMonkey reenvía la información a los usuarios finales

UltraMonkey hace uso del sistema operativo Linux para proporcionar una solución flexible que se pueda adaptar a una amplia gama de necesidades. Desde pequeños clusters formados por dos nodos hasta grandes sistemas que sirven miles de conexiones por segundo.

2.5.1.1 Características del UltraMonkey

Las siguientes características es lo que UltraMonkey ofrece y serán explicadas con más detalle en los siguientes subtemas. Por el momento, con tenerlas presentes es suficiente.

- Balanceo de carga utilizando Linux Virtual Server (LVS) y ldirector
- Alta disponibilidad, el cual incluye el Heartbeat
- Proporciona escalabilidad
- Todo el código es de código abierto (Open Source)

2.5.2 LVS (Linux Virtual Server)[*20]

Virtual Server brinda escalabilidad y un servidor altamente disponible construido sobre un cluster de servidores reales. La arquitectura del cluster de servidores es totalmente transparente a los

[*20] Información recopilada de los sitios oficiales [18] y [19]; y de la página Web http://www.sergio-gonzalez.com/doc/09-conceptos-de-clustering/html/clusster_alta_disponibilidad.html

usuarios finales, y los usuarios interactúan con el cluster como si fuera uno solo. LVS permite balancear las conexiones realizadas por TCP y UDP, permite realizar balanceo de carga sobre el cluster, de tal forma que un nodo se encarga de administrar y repartir las conexiones (nodo master LVS) entre todos los nodos esclavos.

Los servidores reales y el balanceador de carga pueden estar interconectados en una red de área local o geográficamente dispersos en una red WAN (World Area Network). El sistema que tiene instalado LVS es denominado Linux-Director o balanceador de carga, los balanceadores de carga pueden enviar peticiones a los diferentes servidores reales. Brinda escalabilidad en el sistema de forma transparente agregando o removiendo nodos en el cluster. La alta disponibilidad se provee detectando fallas y reconfigurando el sistema de forma apropiada. LVS implementa un programa de espacio de usuario denominado "ipvsadm", el cual nos permite visualizar el status de los nodos en el cluster.

LVS funciona a nivel TCP/IP, es decir, permite encaminar los paquetes entre las redes, realizando por tanto las funciones de encaminamiento o routing. Lo que ve LVS son direcciones y puertos de origen y destino, y toma decisiones para balancear la carga con esta información. LVS toma las decisiones cuando se abre una conexión, manteniendo una tabla de conexiones, para saber a que servidor real enviar un paquete perteneciente a una conexión ya establecida. Por lo tanto, el balanceo de carga que realiza LVS tiene, en principio, granularidad (riqueza funcional) a nivel de conexión.

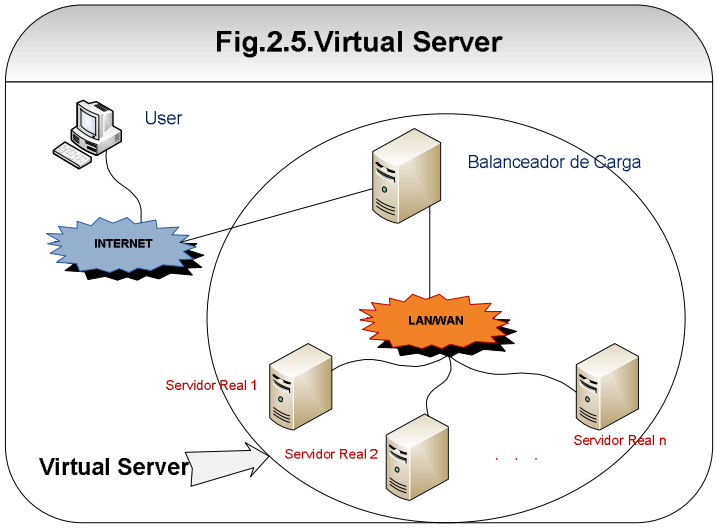
LVS permite balancear muchos protocolos distintos, en principio puede balancear cualquier protocolo que trabaje en un solo puerto, y puede trabajar con protocolos que usen varios puertos, mediante persistencia o marcas de firewall.

Cuando se usan servicios persistentes, cada entrada en la tabla de LVS ya no corresponde a una conexión TCP (direcciones y puertos de origen y destino), sino que sólo usa las direcciones para identificar una conexión (se pierde granularidad).

Se puede usar el firewall iptables o ipchains para marcar los paquetes pertenecientes a un servicio virtual (con una marca de firewall) y usar esa marca para que LVS identifique los paquetes pertenecientes al servicio virtual.

LVS realiza balanceo de carga y facilita la Alta Disponibilidad entre los servidores reales (si alguno deja de funcionar, se elimina del cluster mediante ipvsadm; cuando vuelva a estar operativo, se añade de nuevo con ipvsadm). Sin embargo, el balanceador de carga pasa a ser un SPOF, si se quiere alta disponibilidad se tiene que añadir un balanceador de respaldo y usar software de alta disponibilidad que le permita tomar el papel del balanceador de carga principal, esto lo conseguimos con Heartbeat.

El Virtual Server es un servidor altamente escalable y altamente disponible construido sobre un clúster de servidores verdaderos. La arquitectura del cluster es completamente transparente para los usuarios finales; los usuarios interactúan con el sistema como si fuera un solo servidor de alto rendimiento **Fig.2.5.**



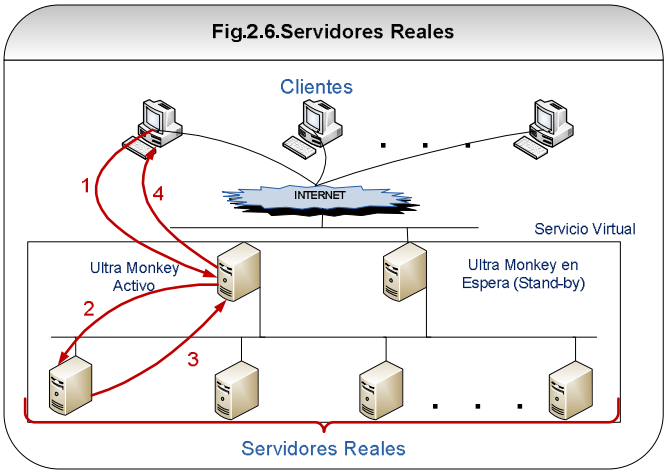
Los servidores reales y los balanceadores de carga pueden estar interconectados por una LAN de alta velocidad o geográficamente dispersado por una WAN.

La escalabilidad del sistema se alcanza agregando o quitando nodos en el clúster. La alta disponibilidad es proporcionada detectando fallas del sistema y reconfigurándolo apropiadamente mientras otro nodo toma su lugar.

El objetivo principal del proyecto Virtual Server de Linux es construir un servidor de alto rendimiento y que cuente con alta disponibilidad en plataformas Linux usando la tecnología "clustering", el cual proporciona escalabilidad, confiabilidad y servicio.

2.5.2.1 Servidores reales

Los servidores reales es el último destino al que llega una conexión hecha a un servicio virtual. Los servidores reales corren las aplicaciones que solicitan los usuarios finales. Los servidores reales están definidos por una dirección IP, un puerto y quizá por una prioridad en caso de que esté definida por el algorítmo en función. Por lo general el puerto de los servidores-reales debe ser igual que el del servicio virtual. El funcionamiento de los servidores-reales se puede supervisar usando ldirectord.



2.5.2.2 Ipvsadm

Ipvsadm es una utilidad para administrar los servicios del Virtual Server ofrecidos por el kernel de Linux con la ayuda de una IP virtual.

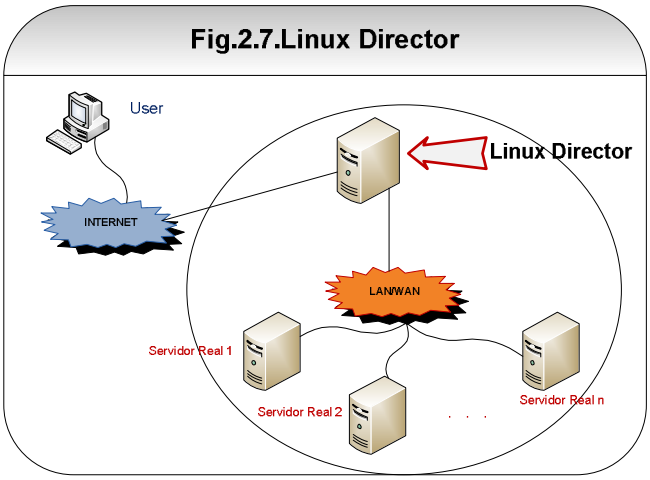
Ipvsadm también nos va a permitir modificar en cualquier momento los servicios y servidores utilizados, esto puede sernos útil para mantenimiento de los servidores.

2.5.2.3 Linux-Director

Le llamamos Linux-Director a la máquina que corre el LVS, el servicio virtual y el servicio que recibe las conexiones. El Linux-Director actúa como un router especializado que reenvía paquetes del usuario final a los servidores reales que en ese momento estén corriendo las aplicaciones.

Cuando las peticiones son recibidas por el Linux-Director un algoritmo decide a cual de los servidores reales enviar el paquete. Una vez tomada esta decisión, los paquetes que sean recibidos de esta conexión, serán enviados al mismo servidor real. De esta manera la integridad de la conexión se mantiene. De forma opcional, un servicio se puede marcar como persistente. Cuando se hace esto, todas las conexiones subsecuentes de un usuario final dado serán enviadas al mismo servidor real hasta que ocurra un lapso de tiempo fuera. Esto se suele utilizar para aplicaciones tales como FTP, donde los usuarios finales deben ser enviados constantemente al mismo servidor real asegurando así la integridad de las conexiones de los datos.

Fig.2.7.Linux Director



2.5.2.4 Servicios Virtuales

En el Linux-Director un servicio virtual es definido por una dirección IP pública, el puerto y el protocolo:

- Una dirección IP: La dirección IP que los usuarios finales utilizarán para accesar al servicio.
- Un puerto: El puerto por el cual los usuarios finales se conectarán.
- Un protocolo: Cualesquiera, UDP o TCP

2.5.2.5 Algoritmo[*21]

El servicio virtual trabaja con un algoritmo que asigna las conexiones entrantes a los servidores reales. Existen muchos algorítmos disponibles y estos pueden ser definidos como parámetros con el comando ipvsadm

ipvsadm -s método_algoritmo

donde método-algorítmo puede ser:

- lc – Least-Connection: Con este algoritmo las peticiones se enviarán al servidor que menos conexiones esté sirviendo en ese momento.

  Si la capacidad de procesamiento de los servidores es similar este algoritmo distribuirá la carga de forma óptima entre todas las máquinas del cluster. Sin embargo si las capacidades de procesamiento varían mucho, la carga no será repartida de forma ecuánime ya que la carga se repartirá según el número de conexiones abiertas en ese momento y no sobre la carga real de cada máquina.

- wlc - Weighted Least-Connection: Asigna más conexiones a los servidores con menos trabajo también tomando en cuenta la prioridad de los servidores reales; es decir, a cada servidor se le asigna una prioridad según su capacidad de procesamiento y aquellos que más prioridad tengan asignada atenderán más peticiones, es decir tendrán más conexiones abiertas.

- rr - Roound Robin: Distribuye el trabajo de igual forma entre los servidores reales disponibles. Este algoritmo es el más simple y lo que hace es distribuir las peticiones entre los servidores de tal manera que si hay 5 servidores y 100 peticiones, cada servidor atenderá a 20 peticiones.

  El orden de distribución de la carga será secuencial, primera petición hacía el primer servidor, segunda al segundo, ..., quinta al quinto, sexta al primero, ...

  Esta distribución es muy sencilla pero presupone que todas las peticiones van a ser equivalentes, en términos de carga, para el servidor, algo que en la realidad dista mucho de ser cierto. O que la capacidad de procesamiento de los servidores es la misma. Podría darse el caso de que haya servidores atendiendo varias peticiones y otros esperando o que los servidores más lentos estuvieran muy sobrecargados mientras que los más potentes estuvieran más desahogados.

- wrr - Weighted Round Robin: Distribuye el trabajo de manera que toma en cuenta la prioridad de los servidores-reales. Los servidores con mayor prioridad es quien recibe primero el trabajo, estos servidores reciben más trabajo que los demás. Los servidores con igual prioridad reciben igual distribución de trabajo.

- lblc - Locality-Based Least-Connection: Asigna los trabajos destinados de una misma IP al mismo servidor-real si este no está sobrecargado y disponible; de lo contrario, los asigna a los servidores con poco trabajo, y lo guarda para una futura asignación.

[*21] Información obtenida del manual del comando ipvsadm: man ipvsadm

2.5.2.6 Balanceo de Carga

Con el crecimiento que se ha venido dando en los últimos años, el tráfico en la red ha aumentado de forma radical y con él, la carga de trabajo que ha de ser soportada por los servidores, especialmente por los servidores Web. Ahora, se hace presente una serie de requerimientos computacionales, de comunicación y de almacenamiento que se deben soportar, esto nos lleva a considerar dos soluciones: la primera, que el servidor se base en una máquina de altas prestaciones, que a largo plazo probablemente vuelva a quedar obsoleta por el crecimiento de la carga o que por los costos que implica sea difícil adquirir; la segunda, encamina la solución a la utilización de la tecnología de clustering para mantener un cluster de servidores, esto se puede llevar a cabo con los recursos computacionales con los que la empresa cuente. Cuando la carga de trabajo crezca, es posible añadir más servidores al clúster[21].

Utilizar un balanceador de carga para distribuir ésta entre los servidores de un cluster es una buena solución. El balanceo de carga puede hacerse a nivel de aplicación y a nivel IP. Cada solución debe elegirse en función de las necesidades del problema al que hacemos frente; mencionemos que Linux Virtual Server (LVS) utiliza balanceo a nivel IP.

El Linux Virtual Server, tiene tres formas de realizar balanceo de carga:

- Mediante NAT
- IP Tunneling
- Direct Routing

2.5.2.6.1 Virtual Server mediante NAT

NAT (*Network Address Translation*) es una técnica utilizada para que una máquina reciba información dirigida a otra y esta pueda reenviarla a quien la solicitó inicialmente. Para ello la máquina que recibe la información, en forma de paquetes, deberá reescribir los paquetes sustituyendo su propia dirección con la de la máquina que realizó la petición (se hace referencia tanto a direcciones físicas, MAC, como lógicas, IP) una vez reescrito el paquete de la forma correcta el balanceador se encargará de enviar los paquetes por la interface adecuada para que le lleguen al destino verdadero del paquete.

Cuando el balanceador reciba peticiones, éste sobrescribirá el paquete y pondrá la dirección de un servidor real, gracias a esto los servidores reales podrán estar ejecutando cualquier sistema operativo que sea accesible por TCP/IP.

Cuando el servidor responda lo hará al balanceador y este reescribirá el paquete, otra vez, poniendo en los paquetes la dirección del cliente que solicitó la información. El balanceador guardará los datos de todas las conexiones que balancee para luego devolver la respuesta al cliente adecuado. Pero no todo son ventajas, esta sobre escritura de los paquetes trae consigo una carga de CPU que puede llegar a ser un cuello de botella. Además tendremos que tener en cuenta cual es el ancho real de nuestra interfaz de red y tener presente que por el balanceador van a pasar tanto las peticiones hacia los servidores, como las respuestas de los servidores hacia los clientes.

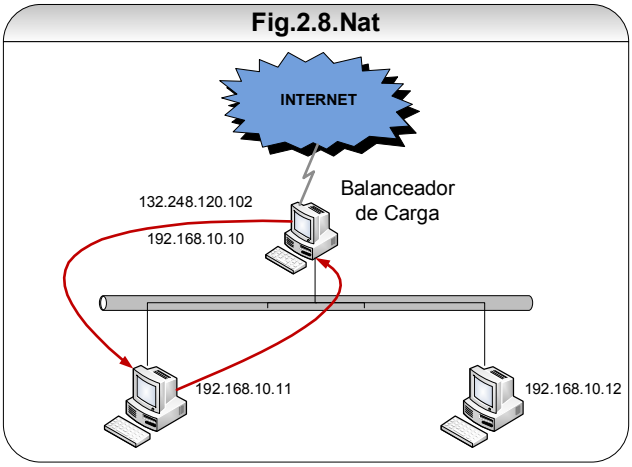
Todos estos paquetes tendrán que ser reescritos por el balanceador y aunque aumentemos la memoria o las capacidades de procesamiento del balanceador todavía estaremos limitados por el ancho real de la interface ya que las respuestas de los servidores ocuparán mas ancho de banda que las peticiones.

---

[21] basado en "utilización y administración avanzada de sistemas GNU/LINUX y aplicaciones, clustering y alta disponibilidad en gnu/Linux", Jose Angel de Bustos

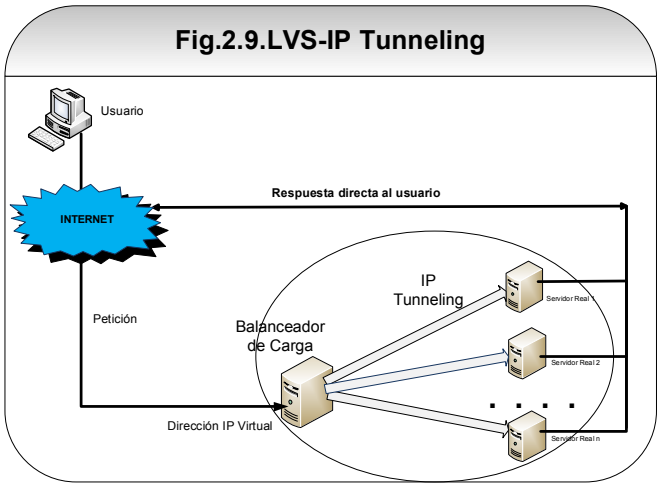
El balanceador deberá tener dos IP una de cara a los posibles clientes a la que llamaremos DIP y otra en la red de los servidores, que llamaremos VIP, es decir que el balanceador deberá hacer funciones de enrutado con lo cual el núcleo deberá estar configurado para ello y tendremos que tener el enrutado habilitado y el núcleo tendrá que estar compilado para poder sobrescribir paquetes.

Los servidores en este caso estarán en la misma red de *VIP* y tendrán como gateway al balanceador de carga.



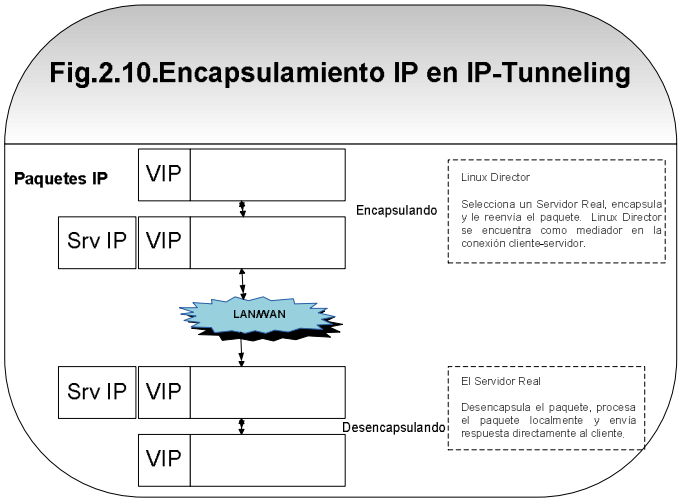
2.5.2.6.2 Virtual Server mediante IP Tunneling

Utilizando IP Tunneling el balanceador únicamente tendrá que hacerse cargo de las peticiones de los clientes y enviarlas a los servidores, siendo estos mismos los que responderán a los clientes. De esta forma el balanceador de carga puede manejar más nodos, es decir el servicio cuenta con mayor escalablilidad.



Una vez que el balanceador de carga tiene el paquete, determina si pertenece a uno de los servicios que tiene balanceados. De ser así encapsula el paquete en otro paquete y se lo envía al servidor de destino. Es este el que se encarga de responder al cliente directamente sin pasar por el balanceador.

El balanceador guarda una tabla de conexiones y cuando le llega un paquete determina si ya existe una conexión abierta y de ser así que servidor real es el que está sirviéndola para enviarle el paquete.



Los servidores deberán estar configurados para trabajar con IP Tunneling (encapsulación) ya que cuando el balanceador recibe un paquete para uno de los servidores éste lo encapsula en un datagrama IP y lo manda a uno de los servidores. Cuando el servidor lo recibe tendrá que desencapsularlo y responderá directamente al cliente sin pasar por el balanceador con lo cual los servidores tendrán que estar conectados tanto al balanceador como a los clientes.

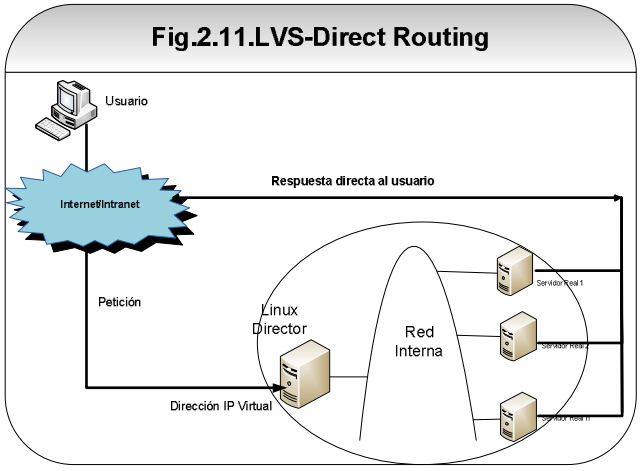
No es necesario que los servidores estén en la misma red, pueden estar geográficamente distribuidos.

En esta configuración surge el problema de ARP.

2.5.2.6.3 Virtual Server mediante Direct Routing

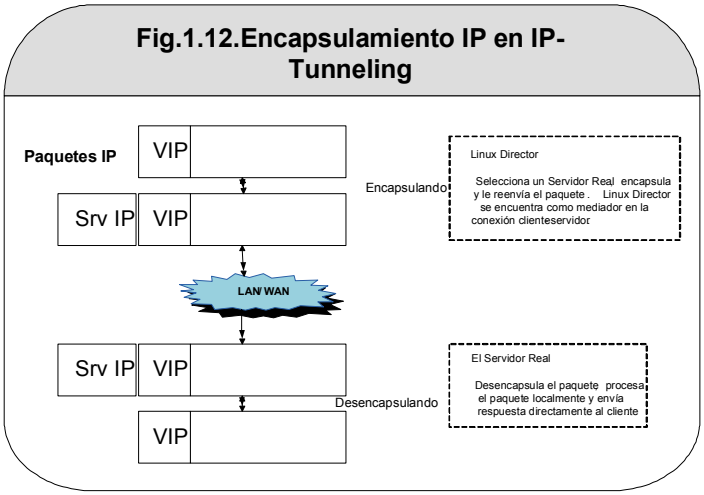
Al igual que en IP Tunneling el balanceador sólo gestionará las peticiones del cliente hacía el servidor, con esto contamos con una solución altamente escalable.

La dirección virtual (VIP) es compartida por el balanceador y los servidores. De esta manera el balanceador recibe las peticiones y las envía a los servidores que procesan las peticiones y dan servicio directamente a los clientes.



En esta solución es necesario que una de las interfaces del balanceador y los servidores estén en el mismo segmento físico de red ya que el balanceador de carga cambiará su dirección física, MAC, en la trama por la dirección física de uno de los servidores que tendrá un alias con la dirección VIP.

El balanceador guarda una tabla de conexiones y cuando le llega un paquete determina si ya existe una conexión abierta y de ser así que servidor real es el que está sirviéndola para enviarle el paquete.



En esta configuración también surge el problema de ARP.

2.5.2.6.4 Problema ARP

Cuando utilizamos Tunneling o Direct Routing tanto el balanceador como los servidores comparten una dirección IP (VIP) y esto puede traer consigo problemas si los balanceadores y los servidores están en la misma red.

El modelo *OSI* consta de 7 capas, las tres primeras son la capa física, la capa de enlace de datos y la capa de red. Cuando un ordenador transmite datos empieza a encapsular en la capa de aplicación, séptima capa. Cuando llega a la capa de red añade la información de red, direcciones IP y direcciones lógicas (origen y destino). Acto seguido añade su dirección física o MAC que es una dirección única que tiene cada tarjeta de red o NIC y que consta de dos partes una que identifica al fabricante de la tarjeta y la otra parte identifica a la tarjeta.

Después en la capa física se traduce toda la información a señales eléctricas y se transmite por el medio. Además de sus direcciones propias IP y MAC se añaden las direcciones del destinatario y si el paquete fuera destinado a una red diferente de la de partida se pondría en el campo de la MAC de destino la MAC de gateway o del router por defecto y este se iría encargando de enrutar el paquete hasta que llegará al último router o gateway el cual cambiaría su MAC por la MAC del equipo que tuviera la IP de destino del paquete. Este último router mandaría el paquete por la interface correspondiente y todos los equipos en esa red desencapsularían el paquete hasta la segunda capa y sólo aquel cuya MAC este en esa trama, como destino, tomará el paquete y lo desencapsulará entero para hacer uso de él.

Cuando utilizamos Tunneling o Direct Routing tenemos que tener en cuenta que los clientes hacen las peticiones al balanceador, y las respuestas las reciben de los servidores.

Tanto el balanceador de carga como los servidores comparten una IP (VIP), cuando un cliente solicita una conexión con VIP la petición se debe de hacer al balanceador, no a los clientes.

Cuando llega una petición de un cliente para el servicio bajo LVS esta llegará desde fuera de la red, con lo cual el router de esa red hará una petición ARP para obtener la MAC de la máquina con la IP VIP.

En esa red podría haber varias máquinas con la IP VIP (el balanceador y los servidores comparten dicha IP) con lo cual cualquiera de ellas podría, y de hecho lo hará, responder a la petición. Pero el paquete deberá ir destinado al balanceador no a los servidores.

El balanceador registrará en sus tablas a que servidor le manda las peticiones y consecuentemente todas las peticiones de ese cliente irán al mismo servidor, bajo la conexión ya establecida. Si uno de los servidores respondiera a la petición ARP el router tendría en su tabla ARP la dirección física del servidor y todos los paquetes se los enviará directamente al servidor sin utilizar el balanceador.

Si en algún momento se cambiara la entrada en la tabla ARP y el router actualizara con la MAC de otra máquina (el balanceador y el resto de servidores tienen una interface o alias con la IP VIP) entonces las peticiones de ese cliente irán a otro servidor en lugar de al servidor que originariamente estaban yendo. Si esto pasa dentro de una misma conexión cuando un servidor empiece a recibir las solicitudes de una conexión que el no ha iniciado (la realizó el servidor que primero respondió a la petición ARP) la conexión se cerrará y habrá que volver a negociarla.

Este problema se presenta con núcleos a partir de la serie 2.2.x y se soluciona haciendo que la interface que tiene la IP VIP no responda a peticiones ARP en los servidores y si en el balanceador de carga. De esta forma nos aseguramos que cuando el router haga una petición ARP para la VIP la única máquina que responda sea el balanceador y de esta forma todos los paquetes para el LVS serán enviados al balanceador y este hará su trabajo.

2.5.2.6.5 Sincronización de conexiones

Para que LVS pueda hacer el envío de todos los paquetes de una conexión al mismo servidor real, la dirección IP del que envía y recibe, puerto de la conexión y el servidor verdadero que está atendiendo esa petición debe ser información con la que cuente el LVS. Si la persistencia está funcionando, entonces la información para permitir que un usuario final se vuelva a conectar al mismo servidor-real se mantiene de manera similar. Si dos Linux-Directors están funcionando de manera activo-pasivo (un Linux-Director activo y el otro en espera), entonces la información de ésta conexión necesita ser sincronizada; de esta manera las conexiones existentes pueden continuar existiendo si el linux-director activo falla y el que se encuentra en espera se activa.

2.5.2.6.6 Ldirectord[*22]

Ldirectord es un demonio que se ejecuta en el maestro LVS y se encarga de testear el servicio de datos realizando una petición y una comprobación para saber si existe una respuesta de los nodos esclavo o servidores reales, y eliminarlos e insertarlos en el cluster dinámicamente, si surge algún problema o si se repone el servicio según sea el caso.

Ldirectord utiliza Heartbeat. monitorea que los servidores reales sigan funcionando periódicamente, enviando una petición a una URL conocida y comprobando que la respuesta contenga una cadena concreta. Si un servidor real falla, entonces el servidor es quitado del conjunto de servidores reales y será reinsertado cuando vuelva a funcionar correctamente. Si todos los servidores fallan, se pude insertar un servidor de fallos, que será quitado una vez que los servidores vuelvan a funcionar. Típicamente, este servidor de fallos es la propia máquina desde el que se realiza el monitoreo.

2.5.3 LINUX-HA Y HEARTBEAT

Linux-HA es un proyecto de código abierto que provee alta disponibilidad. Su componente

[*22] Recopilación de información de la referencia [19], disponible en Internet vía Web URL:http://www.ultramonkey.org/3/ldirectord.html

determinado intervalo de tiempo entre máquinas y principal es Heartbeat el cual implementa el heartbeat-protocol. Esto es, mensajes son enviados en si un mensaje no es recibido de determinada máquina entonces se asume que la máquina ha fallado y alguna medida se toma para corregir este punto.

2.5.3.1 Heartbeat[*23]

Es una herramienta que permite crear un cluster de alta disponibilidad. De momento el cluster sólo soporta 2 nodos, permite crear grupos de recursos y cambiar estos grupos de recursos fácilmente entre nodos. Carece de herramientas de monitorización del servicio de datos, que se consigue con otras herramientas como mon.

Esta tecnología implementa heartbeats, cuya traducción directa sería: «latidos de corazón». Funciona enviando periódicamente un paquete, que si no llegara, indicaría que un servidor no está disponible, por lo tanto se sabe que el servidor ha caído y se toman las medidas necesarias.

Dichos latidos se pueden enviar por una línea serie. De hecho los desarrolladores de HeartBeat recomiendan el uso de puertos serie por varias razones, entre las que destacan que están aislados de las tarjetas de red.

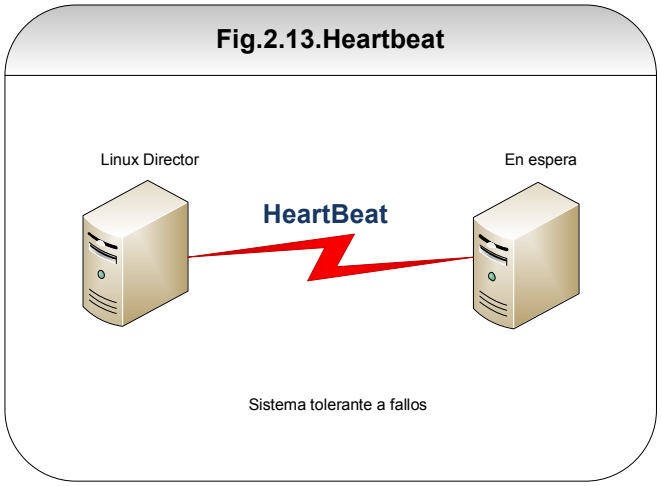
También incluye toma de una dirección IP y un modelo de recursos, incluyendo grupos de recursos. Soporta múltiples direcciones IP y un modelo servidor primario/secundario. Se ha probado satisfactoriamente en varias aplicaciones, como son: servidores DNS, servidores proxy de caché, servidores Web y servidores directores de LVS. El proyecto LVS recomienda HeartBeat para aumentar la disponibilidad de su solución, pero no es parte de LVS.

Cuando HeartBeat es configurado, se selecciona un nodo llamado nodo maestro. Cuando HeartBeat comienza a funcionar, este nodo instala una interface para una dirección IP virtual, es ésta dirección IP la que los usuarios finales accesarán. Si este nodo falla, entonces, otro nodo en el clúster tomará su lugar con la misma IP virtual y utilizando "*gratuitous ARP"* se asegurará de que todo el tráfico enviado a esta dirección sea recibido por la nueva máquina en función. Este método de failover es llamado "*IP Address Takeover"*. Una vez que el nodo principal llegue a estar disponible otra vez los recursos regresarán a formar parte del nodo principal a menos que la bandera auto_failback se encuentre en off en el archivo de configuración ha.cf APÉNDICE 2.

Los mensajes de Heartbeat se envían por todas las líneas de comunicación a la vez, de esta manera, si una línea de apoyo cae, se avisará de ese problema antes de que la línea principal caiga y no haya una línea secundaria para continuar el servicio.

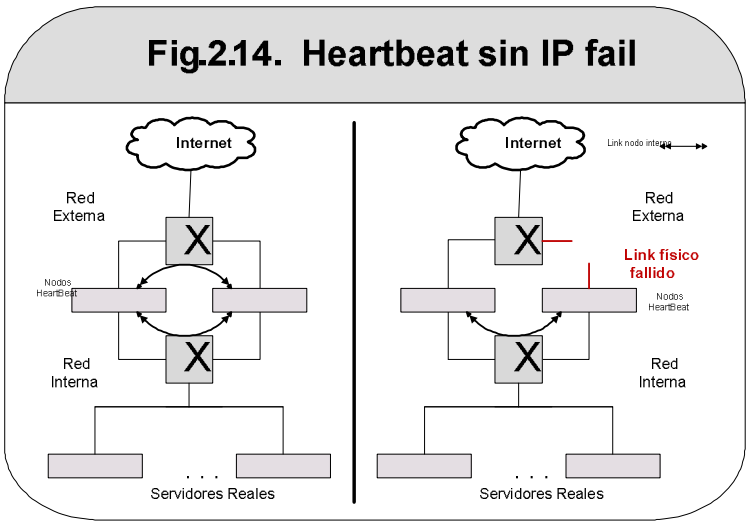
Heartbeat tiene el problema, si no se dispone de una línea dedicada, aunque ésta sea una línea serie, al tener un tráfico que aunque pequeño es constante, suele dar muchas colisiones con otros tráficos que puedan ir por la misma red. Por ejemplo, openMosix y Heartbeat en una misma red que no tenga gran ancho de banda no funcionan bien, sobre todo si hay bastantes nodos, pues los heartbeats se envían de cualquier nodo a cualquier nodo, por lo que podrían llegar a ser un tráfico voluminoso.

[*23] Herramienta indispensable de código abierto en el logro de la alta disponibilidad, disponible en Internet vía Web URL:http://www.sergio-gonzalez.com/doc/09-conceptos-de-clustering/html/clusster_alta_disponibilidad.html



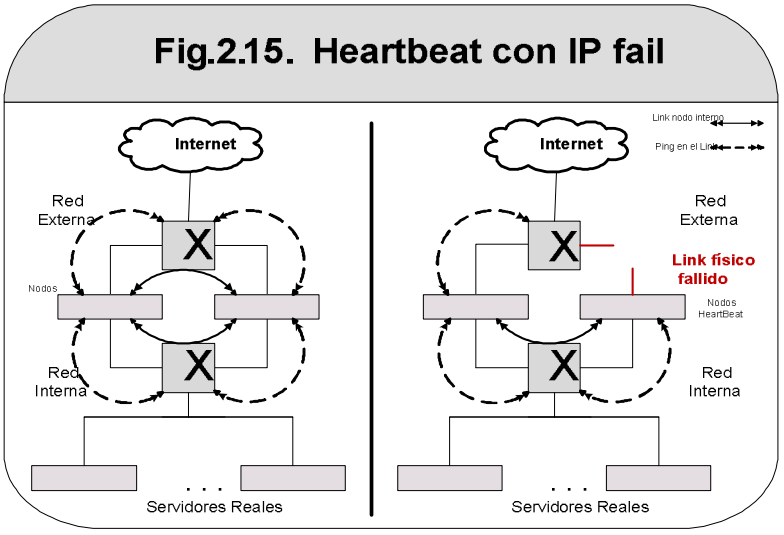
2.5.3.1.1 Ipfail[*24]

El diseño de HeartBeat es tal que si cualquier link de comunicación del nodo está disponible, entonces los nodos se consideran uno a otro como disponibles. Sin embargo, esto no siempre sucede. Por ejemplo, si un par de máquinas tienen links físicos sobre una red interna y una red externa, podría ser propenso a que ocurra un failover si cualquier link falla. Es decir, si dicho estado ocurriera, el nodo no podría direccionar las peticiones entre los usuarios finales y los servidores reales **Fig.2.14.**



El ipfail monitorea uno o más nodos externos conocidos, y verifica el estado de la IP con un método de ping **Fig.2.15.** Por lo regular, éste debería ser un switch o router en la red local. El ping que se hace a éste dispositivo es tratado como un dispositivo quorum. Es decir, como una regla necesaria para que la comunicación entre el servidor y el exterior se cumpla.

[*24] Explicación de heartbeat con Ipfail y sin Ipfail [online]. Disponible en Internet vía Web URL:http://www.ultramonkey.org/3/ipfail.html



2.5.3.1.2 IP Address Takeover [*25]

Si una máquina queda fuera de servicio por cualquier razón, ésta debe ser sustituida por otra máquina. La máquina sustituta es llamada "hot stand-by" (es una máquina que se encuentra en espera de que la otra falle para entrar en servicio de manera inmediata). En el más simple de los casos, IP Adress Takeover involucra dos máquinas, cada una con su propia dirección IP; aunado a lo anterior, existe una dirección IP virtual por la cual accesan los usuarios finales.

La dirección IP virtual que es asignada al servidor maestro. Se le llama IP Address Takeover a la suplantación de la IP virtual del servidor maestro a al servidor sustituto, y está integrado dentro del hearthbeat.

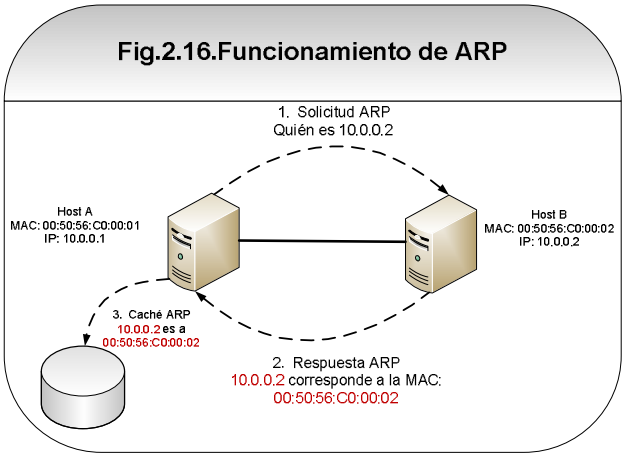
2.5.3.1.3 ARP y la técnica gratuitous ARP

Ip Address Takeover comienza su función cuando el servidor sustituto hace suya la IP virtual, una vez que la IP ha sido añadida, el servidor sustituto está listo para recibir tráfico y contestar peticiones ARP por la dirección IP virtual. Sin embargo, no se asegura que los paquetes sean recibidos por el nuevo servidor, esta técnica es conocida como técnica gratuitous ARP.

La técnica ARP trabaja de la siguiente manera:

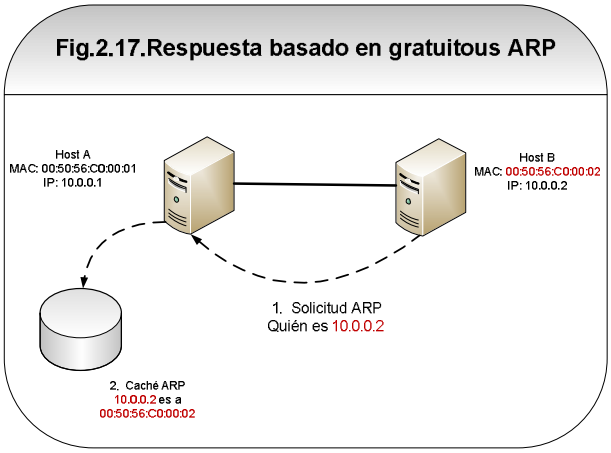
La máquina A envía una petición ARP (solicita saber la MAC correspondiente a una IP) a la máquina B. La máquina B recibe la petición y envía una respuesta, su dirección MAC. La máquina A graba la dirección MAC en su caché ARP, de esta manera no tendrá que estar haciendo la petición ARP y esperar la respuesta cada que quiera enviar un paquete. Sin embargo, la caché ARP típicamente expira después de dos minutos.

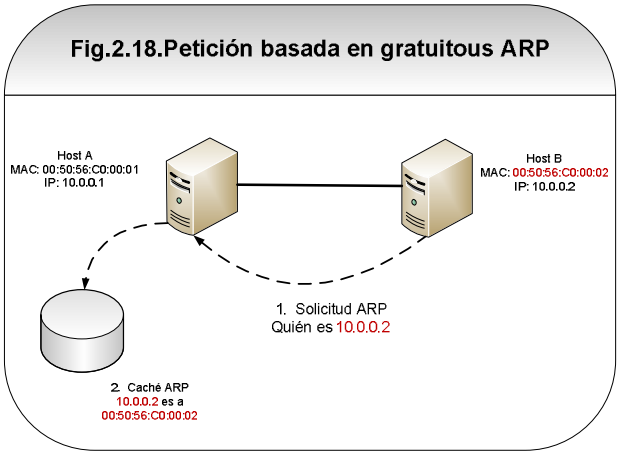
[*25] Explicación del subtema IP Address Takeover [online]. Disponible en Internet vía Web. URL:http://ultramonkey.org/2.0.1/ip_address_takeover.html



Un gratuitous ARP es una respuesta cuando no hay peticiones ARP. Si la respuesta ARP es enviada a un broadcast entonces, todos los host en esa LAN recibirán la respuesta ARP y refrescarán su caché ARP. De igual forma, si una de las máquinas se envía a sí mismo una petición ARP, entonces todos los host de la LAN refrescarán su caché ARP tomando en cuenta la nueva información, la dirección MAC del host que realizó la petición ARP. Ambos tipos de paquetes se envían con el objeto de maximizar el número de host enterados de la noticia.

Es necesario tomar en cuenta que muy poco hardware acepta paquetes de estos dos métodos a menos que le sea indicado en la configuración.





2.5.3.1.4 Seguridad Heartbeat

Heartbeat también se preocupa por la seguridad, permitiendo firmar los paquetes con CRC de 32 bits, MD5 y SHA1. Esto puede evitar el desastre que podría provocarse si un nodo no miembro se enmascarase como nodo miembro del cluster. El problema es que el entorno donde se ejecuta Heartbeat no debe parar nunca y con suerte ese entorno se mantendrá comunicado y funcionando durante años.

Descripción de CRC

CRC (Códigos de Redundancia Cíclica) de 32 bits también llamados códigos cíclicos o códigos polinómicos. Su uso está muy extendido porque pueden implementarse en hardware con mucha facilidad y son muy potentes. Estos códigos se basan en el uso de un polinomio generador $G(x)$ de grado r, y en el principio de que n bits de datos binarios se pueden considerar como los coeficientes de un polinomio de orden $n-1$.

Por ejemplo, los datos 10111 pueden tratarse como el polinomio $x^4 + x^2 + x^1 + x^0$.

A estos bits de datos se le añaden r bits de redundancia de forma que el polinomio resultante sea divisible por el polinomio generador. El receptor verificará si el polinomio recibido es divisible por $G(x)$. Si no lo es, habrá un error en la transmisión.

Los bits de datos se dividen en bloques, y a cada bloque se le calcula r, que se denomina secuencia de comprobación de bloque (Frame Check Sequence, FCS)

Los polinomios generadores más usados son:

CRC-12: $x^{12} + x^{11} + x^3 + x^2 + x^1 + 1$

Usado para transmitir flujos de 6 bits, junto a otros 12 de redundancia. Es decir, usa bloques de 6 bits, a los que les une un FCS que genera de 12 bits.

CRC-16: $x^{16} + x^{15} + x^2 + 1$

Para flujos de 8 bits, con 16 de redundancia. Usado en Estados Unidos, principalmente.
CRC-CCITT: $x^{16} + x^{12} + x^5 + 1$. Para flujos de 8 bits, con 16 de redundancia. Usado en Europa, principalmente.
CRC-32: $x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$

Da una protección extra sobre la que dan los CRC de 16 bits, que suelen dar la suficiente. Se emplea por el comité de estándares de redes locales (IEEE 802) y en algunas aplicaciones del Departamento de Defensa de Estados Unidos.

Si tenemos un polinomio generador tal que $G(x) = x^2 + 1$ podremos obtener a partir de él que el CRC es capaz de detectar todos los errores impares, genera una redundancia de 2 bits y no será capaz de detectar todos los errores dobles, por ejemplo una secuencia error-válido pasaría siempre inadvertida.

Descripción del algoritmo MD5

En criptografía, MD5 (Message-Digest Algorithm 5, Algoritmo de Resumen del Mensaje 5) es un

algoritmo de reducción criptográfico de 128 bits ampliamente usado.

Empezamos suponiendo que tenemos un mensaje de b bits de entrada, y que nos gustaría encontrar su resumen. Aquí b es un valor arbitrario entero no negativo, pero puede ser cero, no tiene por qué ser múltiplo de ocho, y puede ser muy largo. Imaginemos los bits del mensaje escritos así:

$$m0, m1, ..., m\{b-1\}$$

Los siguientes cinco pasos son efectuados para calcular el resumen del mensaje.

Paso 1. Añadiendo bits

El mensaje será extendido hasta que su longitud en bits sea congruente con 448 modulo 512. Esto es, el mensaje se extenderá hasta que se forme el menor número múltiplo de 512 bits. Esta extensión se realiza siempre, incluso si la longitud del mensaje es ya congruente con 448 modulo 512.

La extensión se realiza como sigue: un sólo bit "1" se añade al mensaje, y después bits "0" se añaden hasta que la longitud en bits del mensaje extendido se haga congruente con 448 modulo 512. En todos los mensajes se añade al menos un bit y como máximo 512.

Paso 2. Longitud del mensaje

Una representación de 64 bits de b (la longitud del mensaje antes de añadir los bits) se concatena al resultado del paso anterior. En el supuesto no deseado de que b sea mayor que 2^{64}, entonces sólo los 64 bits de menor peso de b se usarán.

En este punto el mensaje resultante, después de rellenar con los bits y con b, se tiene una longitud que es un múltiplo exacto de 512 bits. A su vez, la longitud del mensaje es múltiplo de 16 palabras (32 bits por palabra). Con $M[0...N-1]$ denotaremos las palabras del mensaje resultante, donde N es múltiplo de 16.

Paso 3. Inicializar el búfer MD

Un búfer de cuatro palabras (A, B, C, D) se usa para calcular el resumen del mensaje. Aquí cada una de las letras A, B, C, D representa un registro de 32 bits. Estos registros se inicializan con los siguientes valores hexadecimales, los bits de menor peso primero:

palabra A: 01 23 45 67
palabra B: 89 *ab cd ef*
palabra C: *fe dc ba* 98
palabra D: 76 54 32 10

Paso 4. Procesado del mensaje en bloques de 16 palabras

Primero definimos cuatro funciones auxiliares que toman como entrada tres palabras de 32 bits y su salida es una palabra de 32 bits. Los operadores son las funciones XOR, AND, OR y NOT respectivamente. En cada posición de cada bit F actúa como un condicional: si X, entonces Y sino Z. Es interesante resaltar que si los bits de X, Y, Z son independientes y no sesgados, cada uno de los bits de $F(X, Y, Z)$ será independiente y no sesgado.

Las funciones G,H,I son similares a la función F, ya que actúan "bit a bit en paralelo" para producir sus salidas de los bits de X,Y,Z, en la medida que si cada bit correspondiente de X,Y,Z son independientes y no sesgados, entonces cada bit de $G(X,Y,Z),H(X,Y,Z),I(X,Y,Z)$ serán independientes y no sesgados. Nótese que la función H es la comparación bit a bit "xor" o función "paridad" de sus entradas.

Este paso usa una tabla de 64 elementos $T[1...64]$ construida con la función $Seno$. Denotaremos por $T[i]$ el elemento i-ésimo de esta tabla, que será igual a la parte entera del valor absoluto de $4294967296 * Sen(i)$, donde 'i' está en radianes.

Paso 5. Salida

El resumen del mensaje es la salida producida por A,B,C,D. Esto es, se comienza el byte de menor peso de A y se acaba con el byte de mayor peso de D.

Descripción de SHA

La familia SHA (Secure Hash Algorithm, Algoritmo de Hash Seguro) es un sistema de funciones hash criptográficas relacionadas de la Agencia de Seguridad Nacional de los Estados Unidos y publicadas por el National Institute of Standards and Technology (NIST).

SHA-1 ha sido examinado muy de cerca por la comunidad criptográfica pública, y no se ha encontrado ningún ataque efectivo. No obstante, en el año 2004, un número de ataques significativos fueron divulgados sobre funciones criptográficas de hash con una estructura similar a SHA-1; esto ha planteado dudas sobre la seguridad a largo plazo de SHA-1.

SHA-0 y SHA-1 producen una salida resumen de 160 bits de un mensaje que puede tener un tamaño máximo de 2^{64} bits, y se basa en principios similares a los usados por el profesor Ronald L. Rivest del MIT en el diseño de los algoritmos de resumen del mensaje MD4 y MD5.

Hay varias operaciones de mantenimiento de seguridad que necesitan ser efectuadas en ese tiempo, como pueden ser cambio de claves y de protocolos de autentificación. Heartbeat está preparado para esos cambios disponiendo de ficheros para la configuración.

2.6 DEMONIO DE SERVICIO DE MONITOREO MON

MON es una herramienta de monitoreo de disponibilidad de los servicios el cual envía alertas sobre eventos predefinidos en su archivo de configuración. Los servicios son definidos y monitoreados por un programa, el cual puede ser tan simple como realizar ping al sistema o tan complejo como el análisis de los resultados de una transacción cualquiera. Las alertas son acciones como el envío de un correo electrónico o las acciones necesarias para que en alta disponibilidad se cedan los derechos del manejo del servicio al servidor en stand-by.

Una meta en el diseño del MON es mantener simplicidad de modo que el sistema pueda ser escalable, fácil de utilizar y extender para ampliar su variedad de uso.

La curva de aprendizaje es muy baja para la instalación, la configuración, y el arreglo para requisitos particulares iniciales. Los monitores y las alarmas son simples de escribir con cualquier lenguaje, y simple de incorporar en una configuración de cualquier site.

El reporte y el control de la funcionalidad es fácil de personalizar con la ayuda de una interfaz

basada en el protocolo TCP.

MON es un software de código abierto y distribuido bajo la licencia de GNU Public versión 2.

La herramienta es extremadamente utilizable por sistemas administradores, pero su uso no está limitado para ello. Fue diseñado para atender problemas de propósito general en el sistema, separando las tareas de verificación de servicios para la disponibilidad y enviando alertas cuando alguna cosa falla.

Para lograrlo, MON es implementado como un scheduler (planificador de horarios) el cual corre los programas o scripts que están siendo testeados, y dispara alertas cuando estos programas o scripts detectan algún fallo.

Ningún servicio se maneja para monitorearse por su mismo servidor. Es obvio que si la máquina llegase a tener algún conflicto, dependiendo del conflicto, será dificil que ésta pueda mandar alertas para dar aviso.

La supervisión de la temperatura en el site se puede llevar a cabo agregando un script que recopile datos de un transistor térmico (termistor) vía puerto serial.

A menudo estos scripts de monitoreo son modelos para realizar el monitoreo de servicios existentes o comandos que nos arrojan información importante acerca del estado de la máquina tales como un "ping" o "ftp".

El scheduler del MON también puede mantener clientes en red permitiendo la manipulación de los parámetros de ejecución, deshabilitando y habilitando las alertas y testeos, listando las fallas y el historial de alertas, y reportando los estados actuales de todos los monitores.

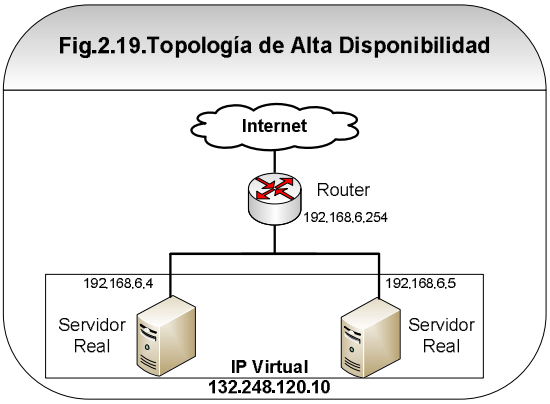
2.7 TOPOLOGIAS[*26]

Las siguientes topologías envuelven todos los conceptos mencionados anteriormente, cada topología muestra una solución diferente; la elección de ésta depende de las necesidades a cubrir y los componentes hardware con los que se cuenta.

2.7.1 Topología de Alta Disponibilidad

Esta topología provee Alta Disponibilidad en el servicio con mínimos requerimientos en hardware.

Fig.2.19.Topología de Alta Disponibilidad



[*26] El proyecto UltraMonkey define las topologías sobre las que funciona [online]. Disponible en Internet vía Web URL:http://www.ultramonkey.org/3/topologies/

Cada servidor real cuenta con una IP no homologada mediante la cual se están monitoreando y el servidor real que está en funcionamiento será quien tenga posesión de la IP virtual. Ambos servidores reales se están monitoreando uno al otro utilizando Heartbeat y en el momento en que el servidor principal falla, el que se encuentra en espera asume las responsabilidades del otro haciendo suya la IP virtual; de esta manera el servicio siempre está disponible.

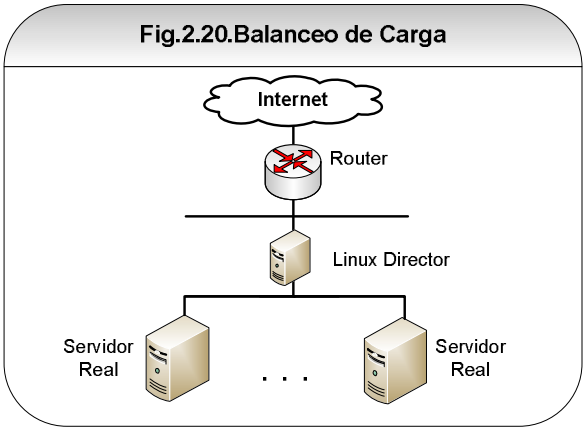
Esta topología únicamente requiere de la configuración del HeartBeat en los servidores reales con el objeto de mantener un monitoreo mutuo.

Los servidores reales pueden ofrecer una gran variedad de servicios incluido el servicio de páginas Web, esto dependerá de las necesidades a satisfacer.

2.7.2 Topología Balanceo de Carga

Esta topología se enfoca en proporcionar balanceo de carga con el mínimo de requerimientos de hardware. Si se requiere, se pueden agregar más servidores reales como recurso adicional.

Fig.2.20.Balanceo de Carga



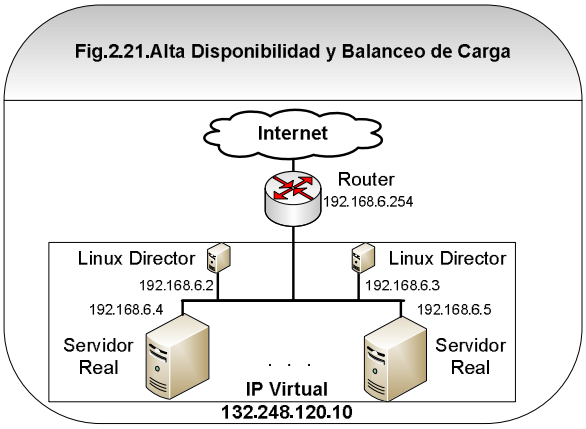
Para esta topología es necesaria la configuración de Linux Director y en él el Ldirector, recordemos que el Linux Director se encarga de balancear el tráfico utilizando LVS.

Cuando el Linux Director recibe conexiones de los usuarios finales, toma la decisión sobre cuál servidor real enviar la petición. Todos los paquetes mientras la conexión se encuentre activa, serán enviados al mismo servidor real, de esta manera la integridad de la conexión se mantiene.

El Ldirector ayuda en el monitoreo del estado de los servidores reales haciendo peticiones de una página conocida y comprobando que contenga una secuencia predeterminada. Si un servidor real falla entonces el servidor es tomado como fuera de servicio y será reinsertado una vez que vuelva a funcionar.

2.7.3 Topología Alta disponibilidad y Balanceo de carga

Esta topología proporciona Alta Disponibilidad y Balanceo de carga. Mínimo, cuatro nodos son requeridos para la implementación de ésta topología. Pueden ser añadidos más servidores reales a la topología de acuerdo a los requerimientos que se soliciten.



El Linux Director activo acepta tráfico de la dirección IP virtual y realiza balanceo de esta carga utilizando LVS. Ambos Linux Director se monitorean utilizando Heartbeat y en el momento en que el Linux Director activo falle, el de reemplazo asume el papel de Linux Director principal y toma posesión de la IP virtual, de esta manera el servicio se mantiene siempre activo. Las conexiones son sincronizadas entre ambos Linux Director, de tal forma que cuando ocurre un failover las conexiones existentes pueden continuar. Cuando el Linux Director recibe conexiones de los usuarios finales, toma la decisión sobre cuál servidor real enviar la petición. Todos los paquetes mientras la conexión se encuentre activa, serán enviados al mismo servidor real, de esta manera la integridad de la conexión se mantiene.

El Ldirector ayuda en el monitoreo del estado de los servidores reales haciendo peticiones de una página conocida y comprobando que contenga una secuencia predeterminada. Si un servidor real falla entonces el servidor es tomado como fuera de servicio y será reinsertado una vez que vuelva a funcionar.

Los servidores reales pueden ofrecer una gran variedad de servicios incluido el servicio de páginas Web, esto dependerá de las necesidades a satisfacer.

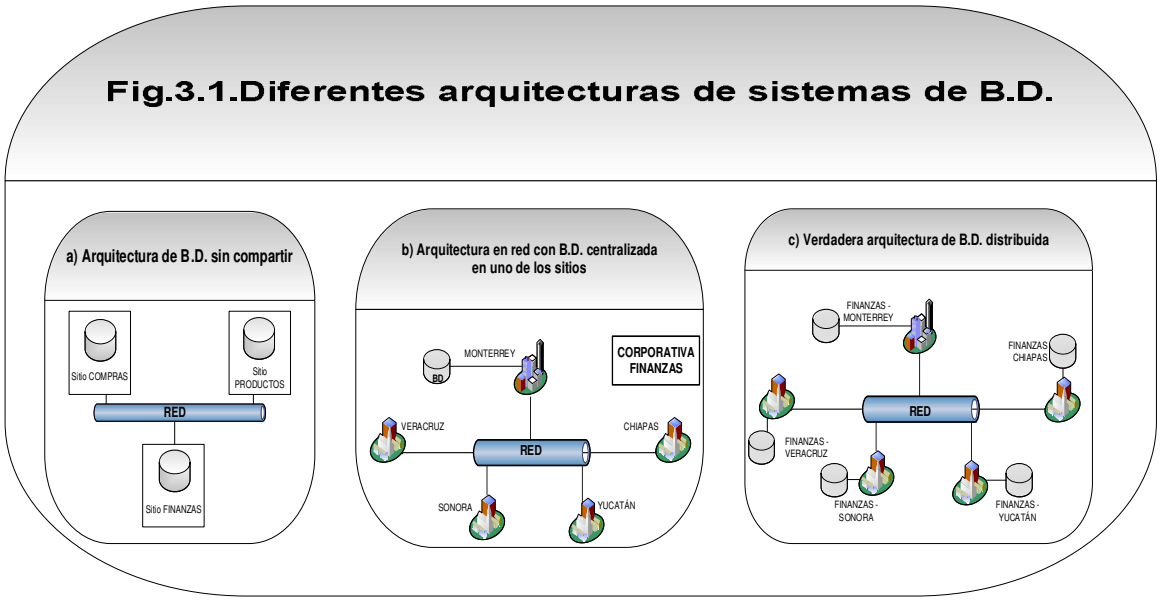
CAPÍTULO III REPLICACIÓN DE DATOS

Recordemos que la aplicación "Tiendas electrónicas UNAM" trabaja con una base de datos en el sistema manejador Sybase, por tanto este capítulo abordará el concepto de Base de Datos Distribuida y de la Replicación de Datos para dar una solución completa al proyecto de la tesis; y como último tema se mencionará todo lo concerniente al Sistema Manejador de Base de Datos Sybase para comprender su funcionamiento y la implementación del Replication Server.

3.1 BASE DE DATOS DISTRIBUIDAS[7]

Un sistema distribuido de base de datos consiste en un conjunto de computadoras, cada una de las cuales puede participar en la ejecución de transacciones que accesen datos de una o varias de ellas. La diferencia principal entre los sistemas de base de datos *centralizados* y *distribuidos* es que, en los primeros, todos los datos residen en un solo servidor, mientras que en los últimos, se encuentran en varios **Fig 3.1.**

Una base de datos distribuida es una colección de datos que pertenece lógicamente al mismo sistema pero que está dispersa físicamente entre los sitios de una red de computadores. Varios factores han dado pie a la creación de los sistemas de base de datos distribuidos (SBDD); un SBDD es un sistema software que maneja una base de datos distribuida haciendo la distribución transparente para el usuario.



3.1.1 Ventajas

La gestión de bases de datos distribuidas se ha propuesto por varias razones, que van desde la descentralización organizacional y procesamiento económico hasta la gran autonomía. Entre las ventajas potenciales de los SBDD están las siguientes:

La naturaleza distribuida de algunas aplicaciones de bases de datos

Muchas de estas aplicaciones están distribuidas naturalmente en diferentes lugares. Por ejemplo, una compañía puede tener oficinas en varias ciudades, o un banco puede tener múltiples sucursales. Es natural que las bases de datos empleadas en tales aplicaciones estén distribuidas en

[7] Ramez Elmasri, Shamkant B. Navathe; Fundamentos de Sistemas de Bases de Datos, capítulo I

esos lugares. Muchos usuarios locales tienen acceso exclusivamente a los datos que están en el lugar, pero otros usuarios globales como la oficina central de la compañía pueden requerir acceso ocasional a los datos almacenados en varios de estos lugares. Cabe señalar que, por lo regular, los datos en cada sitio local describen un "mini mundo" en ese sitio. Las fuentes de los datos y la mayoría de los usuarios y aplicaciones de la base datos local residen físicamente en ese lugar.

Mayor fiabilidad y disponibilidad

Estas son dos de las ventajas potenciales de las bases de datos distribuidas. La *fiabilidad* se define a grandes rasgos como la probabilidad de que un sistema esté en funciones en un momento determinado, y la *disponibilidad* es la probabilidad de que el sistema esté disponible continuamente durante un intervalo de tiempo. Cuando los datos están distribuidos en varios sitios, un sitio puede fallar mientras que los demás siguen operando. Sólo los datos y el software que residen en el sitio que falló están inaccesibles. Esto mejora tanto la fiabilidad como la disponibilidad. Se logran mejoras adicionales si se replican, con un criterio adecuado, los datos y el software en más de un sitio. En un sistema centralizado, el fallo de un solo sitio hace que el sistema completo deje de estar disponible para todos los usuarios.

Posibilidad de compartir los datos al tiempo que se mantiene un cierto grado de control local

En algunos tipos de sistemas de bases de datos distribuidas SBDD, es posible controlar los datos y el software localmente en cada sitio. Sin embargo, los usuarios de otros sitios remotos pueden tener acceso a ciertos datos. Esto hace posible el compartimiento controlado de los datos en todo el sistema distribuido.

Mejor rendimiento

Cuando una base de datos grande está distribuida en múltiples sitios, hay bases más pequeñas en cada uno de éstos. En consecuencia, las consultas locales y las transacciones que tienen acceso a datos de un solo sitio tienen un mejor rendimiento porque las bases de datos locales son más pequeñas. Además, cada sitio tiene un menor número de transacciones en ejecución que si todas las transacciones se enviaran a una sola base de datos centralizada. En el caso de transacciones que impliquen acceso a más de un sitio, el procesamiento en los diferentes sitios puede efectuarse en paralelo., reduciéndose así el tiempo de respuesta.

Consideraciones...

La distribución conlleva un incremento de la complejidad en el diseño e implementación del sistema. Para conseguir las ventajas potenciales enumeradas previamente, el software del sistema de base de datos (sistema software que maneja una BDD haciendo la distribución transparente para el usuario) debe poder proporcionar las siguientes funciones además de las proporcionadas por los sistemas de base de datos centralizados.

- Procesamiento de consultas distribuidas: la capacidad de tener acceso a sitios remotos y transmitir consultas y datos entre los diversos sitios a través de una red de comunicaciones.

- Mantenimiento de la pista de los datos: la capacidad de seguir la pista a la distribución y la replicación de los datos en el catálogo del sistema de base de datos distribuida SBDD.

- Gestión de transacciones distribuidas: la capacidad de elaborar estrategias de ejecución para consultas y transacciones que tienen acceso a datos de más de un sitio.

- ❖ Recuperación de bases de datos distribuidas: la capacidad de recuperarse de caídas de sitios individuales y de nuevos tipos de fallos, como el fallo de un enlace de comunicación.

Estas funciones elevan la complejidad de un sistema de bases de datos distribuidas en comparación con un sistema de bases de datos centralizado.

En el nivel físico, de hardware, los principales factores que distinguen un sistema de bases de datos distribuidas de un sistema centralizado son los siguientes:

- Hay múltiples computadores, llamados sitios o nodos.
- Estos sitios deben estar comunicados por medio de algún tipo de red de comunicaciones para transmitir datos y órdenes entre los sitios como se muestra en la **Fig.3.1.c)**

Los sitios pueden estar cercanos físicamente, digamos, dentro del mismo edificio o grupo de edificios adyacentes y conectados a través de una red de área local, o pueden estar distribuidos geográficamente a grandes distancias y conectados a través de una red de larga distancia. Las redes de área local normalmente utilizan cables, mientras que las redes de larga distancia utilizan líneas telefónicas o satélites. También es posible utilizar una combinación de los dos tipos de redes.

Las redes podrían tener diferentes topologías que definen los caminos de comunicación directos entre sitios. Los tipos y topologías de las redes utilizadas podrían tener un efecto significativo en el rendimiento y por tanto en las estrategias de procesamiento de consultas distribuidas y de diseño de bases de datos distribuidas. Para temas de arquitectura de alto nivel, sin embargo, no importa el tipo de red a utilizar; lo único importante es que cada sitio puede comunicarse, directa e indirectamente, con todos los demás sitios.

3.1.2 Tipos de sistemas de bases de datos distribuidas

El término sistema de base de datos distribuida puede describir diversos sistemas que presentan muchas diferencias entre sí. El punto principal que todos estos sistemas tienen en común es el hecho de que los datos y el software están distribuidos entre múltiples sitios conectados por los criterios y factores que distinguen a algunos de estos sistemas.

El primer factor que consideraremos es el grado de homogeneidad del software del SBDD. Si todos los servidores o (SBD locales individuales) utilizan software idéntico y todos los usuarios (clientes) emplean software idéntico, se dice que el SBDD es homogéneo; en caso contrario se le denomina heterogéneo. Otro factor relacionado con el grado de homogeneidad es el grado de autonomía local. Si el sitio local no puede funcionar como un SBD autónomo, el sistema no tiene autonomía local. Por otro lado, si se permite a las transacciones locales acceso directo a un servidor, el sistema tendrá cierto grado de autonomía local.

En un extremo de la gama de autonomía, tenemos un SBDD que da al usuario la impresión de ser un SBD centralizado. Sólo hay un esquema conceptual, y todo acceso al sistema es a través de un sitio que es parte del SBDD, de modo que no hay autonomía local. En el otro extremo nos encontramos con un tipo de SBDD denominado SBDD federado (o sistema de múltiples bases de datos). En un sistema así, cada servidor es un SBD centralizado independiente y autónomo que tiene sus propios usuarios locales, transacciones locales, y por tanto posee un alto grado de autonomía local. El término sistema de bases de datos federadas (SBDF) se utiliza cuando hay algún esquema o vista global de la federación de bases de datos que es compartido por las aplicaciones. Por otro lado, un sistema de múltiples bases de datos no tiene un esquema global e interactivamente construye uno según la aplicación lo necesite. Los dos sistemas son un híbrido

entre sistemas centralizados y distribuidos, y la distinción que hemos hecho entre ellos no se sigue estrictamente.

3.2 CONCEPTOS GENERALES DE LA REPLICACIÒN DE DATOS[*27]

3.2.1 Concepto de replicación

La replicación consiste en el transporte de datos entre dos o más servidores, permitiendo que ciertos datos de la base de datos (BD) estén almacenados en más de un sitio, y así aumentar su disponibilidad y mejorar el funcionamiento de las consultas globales.

Informalmente, una transacción es la ejecución de ciertas instrucciones u operaciones que modifican ciertos datos y accesan a una BD compartida. Un sistema de replicación de transacciones mantiene consistentes, datos sincronizados en BD separadas, es decir, cuando se realiza alguna transacción en una de las BD (llamada base de datos primaria) y envía las operaciones a otra llamada base de datos replicada. A las operaciones que realizan cambios en los datos, que son capturadas y enviadas, se les llama transacciones replicadas.

La replicación resulta útil para mejorar la disponibilidad de los datos. El caso más extremo es la replicación de toda la base de datos en todos los sitios del sistema distribuido, creando así una base de datos distribuida totalmente replicada. Esto puede mejorar la disponibilidad notablemente porque el sistema puede seguir operando mientras, por lo menos, uno de los sitios esté activo. También mejora el rendimiento de la recuperación en consultas globales, porque el resultado de semejante consulta se puede obtener localmente en cualquier sitio; así, una consulta de recuperación se puede procesar en el sitio local donde se introduce, si dicho sitio cuenta con un módulo servidor. La desventaja de la replicación completa es que puede reducir drásticamente la rapidez de las operaciones de actualización, pues una sola actualización lógica deberá ejecutar en todas y cada una de las copias de la base de datos a fin de mantener la consistencia. Esto es especialmente cierto si existen muchas copias de la base de datos. Con la replicación completa, las técnicas de control de concurrencia y recuperación se vuelven más costosas de lo que serían sin replicación.

El extremo opuesto a la replicación completa es no tener ninguna replicación. En estos dos extremos, tenemos una amplia gama de replicación parcial de los datos; es decir, algunos fragmentos de la base de datos pueden estar replicados y otros no. El número de copias de cada fragmento puede ir desde uno hasta el número total de sitios en el sistema distribuido. Un caso especial de replicación parcial ocurre con frecuencia en aplicaciones donde trabajadores que se desplazan, como vendedores, gestores financieros y reguladores de reclamaciones, llevan con ellos bases de datos parcialmente replicadas en computadores portátiles y asistentes digitales personales y las sincronizan periódicamente con la base de datos en el servidor. En ocasiones se llama esquema de replicación a una descripción de la replicación de los fragmentos.

3.2.2 Ventajas de la replicación de datos

Partamos de que tanto los sistemas de BD centralizadas como los sistemas de BD distribuidas deben ocuparse de los problemas asociados al acceso remoto:

- La respuesta en la red es lenta cuando hay tráfico cargado en la WAN.
- Un servidor de datos centralizados puede convertirse en un embotellamiento si es que una larga comunidad de usuarios intenta accesar al servidor de datos.
- Los datos son inaccesibles cuando ocurre una falla en la red.

[*27] Recopilación de información de los libros [10] capítulo XIV p. 528-530, libro [6].

Los problemas del funcionamiento y la disponibilidad asociados al acceso a la BD pueden ser solucionados realizando una replicación de datos de la BD fuente a la BD local. El Servidor de Réplica debe proporcionar:

- ✓ Costo-efectividad
- ✓ Un sistema de tolerancia a fallos para la replicación de datos.
- ✓ Mantener la integridad de los datos replicados a través del sistema mientras también se incrementa la disponibilidad de éstos.
- ✓ Replicar las invocaciones de los procedimientos almacenados
- ✓ Mantener datos actualizados en las múltiples BD de modo que los clientes puedan acceder a los datos locales en lugar de acceder remotamente a una BD centralizada y de esta manera evitar el embotellamiento

3.2.3 Transacción[*28]

"Las transacciones son un medio para que las operaciones ACID se conviertan en materia prima." [Gray y Reuter]

Desde la perspectiva de un negocio, toda transacción es una actividad que cambia el estado de una empresa; por ejemplo, un cliente que deposita en una cuenta de cheques constituye una transacción bancaria. En términos técnicos, transacción es un conjunto de acciones con las propiedades ACID. En este caso, ACID, término acuñado por Andreas Reuter en 1983, significa atomicidad, congruencia, aislamiento y durabilidad (Atomicity, Consistency, Isolation and Durability.

Atomicidad. Significa que una transacción es una unidad indivisible de trabajo: todas las acciones que comprende cumplen su cometido o no. Es una propuesta de todo o nada. Las actividades que abarcan las transacciones pueden incluir las colas de mensajes, las actualizaciones de una base de datos y la exhibición de los resultados en la pantalla de la máquina cliente. La atomicidad se define desde el punto de vista del consumidor de la transacción.

Congruencia. Significa que después de que se ejecuta una transacción, ésta debe salir del sistema en un estado correcto o ha de abortarse. Si la transacción no puede alcanzar una condición estable, debe restaurar el sistema, dejarlo en su estado original.

Aislamiento. Significa que las transacciones que se ejecutan de forma concurrente no influyen entre sí en el comportamiento de las demás. La transacción debe seriar todos los accesos a los recursos compartidos y garantizar que los programas concurrentes no corromperán las operaciones de las otras. Un programa multiusuario que funcione con protección de transacciones debe comportarse como si estuviera en un entorno de un solo usuario. Los cambios que realiza una transacción en los recursos compartidos no se verán fuera de la transacción hasta que ésta cumpla su cometido.

Durabilidad. Significa que los efectos de una transacción son permanentes una vez que termina su trabajo. Los cambios que realice deben sobrevivir a las fallas del sistema. El término persistente es sinónimo de durable.

Una transacción se convierte en la unidad fundamental de restauración, congruencia y concurrencia de un sistema cliente/servidor. ¿Por qué reviste tanta importancia? Pongamos por caso una sencilla operación bancaria débito-crédito. Sin duda, a usted le gustaría ver que todo el crédito otorgado a su

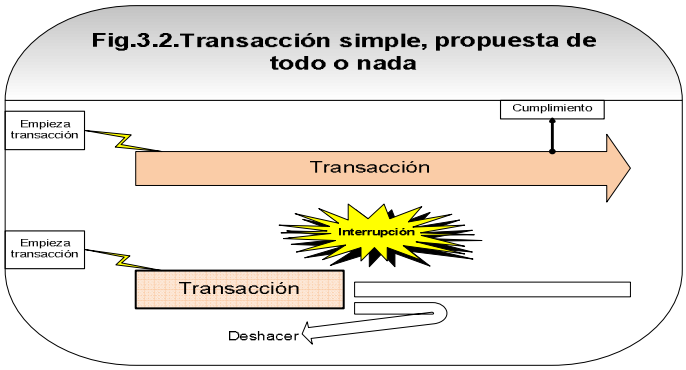
[*28] Robert Orfali hace referencia a las propiedades ACID, bibliografía [6].

cuenta fue un éxito. Cualquier pérdida sería inaceptable (naturalmente, el crédito que se le otorgue sin haberlo solicitado es bienvenido). Esto significa que usted depende de la aplicación para recibir la cantidad esperada en una transacción de negocios del mundo real. A su vez, la aplicación depende del sistema básico, por lo general un monitor TP, para alcanzar tal grado de integridad de la transacción. El programador no tendría que escribir toneladas de código para reinventar la rueda de la transacción.

Un aspecto aún más sutil es que todos los programas que participan deben ajustarse a la disciplina transaccional, pues con uno que no lo haga puede corromperse todo el sistema. Una transacción que desde el principio use, de manera inadvertida, datos corrompidos, producidos por un programa no transaccional se erigiría sobre un fundamento cenagoso.

¿Cuándo debe comenzar una transacción? ¿Cuándo ha de terminar y de poner sus efectos a disposición del mundo? ¿Cuáles son las unidades de restauración apropiadas si ocurren fallas?

Las transacciones simples resultan simples porque todo el trabajo realizado dentro de los límites de una transacción ocurre en el mismo nivel. La transacción comienza con la instrucción de arranque (begin_transaction, por ejemplo) y termina con la de transacción cumplida (commit_transaction, por decir algo) o con la de abortar transacción (digamos, abort_transaction). Es un planteamiento de todo o nada: no hay modo de cumplir o de abortar partes de una transacción simple. Ninguna actividad puede dividirse, que es lo que se quiere antes que cualquier otra cosa. En el siguiente cuadro se comparan los comandos empleados en diferentes monitores de TP para marcar los límites de una transacción.



Las transacciones existen para ayudarnos a simplificar nuestras aplicaciones y darnos mejor control del entorno donde se ejecutan. Algunas extensiones propuestas pueden ocasionar más problemas de los que resuelven. En todo caso, no importa qué extensiones demuestren ser las más útiles e importantes en el futuro, las transacciones simples seguirán representando el corazón de todos los mecanismos indispensables para que los modelos más poderosos funcionen. La mayor parte de las bases de datos, MOM, monitores de TP y monitores de transacciones de objetos (OTM, object transaction monitors) implementan el modelo de las transacciones simples.

3.2.4 Componentes del modelo de replicación

Para representar los componentes y procesos de una topología de replicación se utilizará un modelo, el cual se compone de los siguientes objetos: el publicador, el distribuidor y el suscriptor.

La replicación de datos es un asunto exclusivamente entre servidores de datos, en nuestro caso

hablamos de servidores SQL Server. Los servidores SQL Server pueden desempeñar uno o varios de los siguientes roles: publicador, distribuidor o suscriptor.

El publicador

Es un servidor que pone los datos a disposición de otros servidores para poder replicarlos. El publicador, además, detecta qué datos han cambiado durante la replicación transaccional y mantiene información acerca de todas las publicaciones del sitio.

El distribuidor

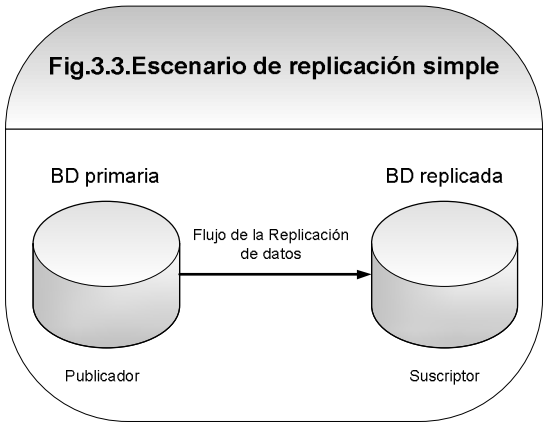
Es un servidor que aloja la base de datos de distribución y almacena los datos históricos, transacciones y meta datos. La función del distribuidor varía según la metodología de replicación implementada. En ocasiones se configura como distribuidor el mismo publicador y se le denomina *distribuidor local*. En el resto de los casos el distribuidor será remoto, pudiendo coincidir en algún caso con un suscriptor.

Los suscriptores

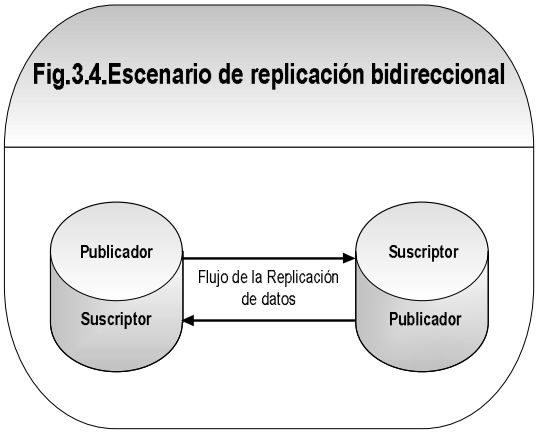
Reciben los datos replicados. Los suscriptores además de obtener sus suscripciones, en dependencia del tipo y opciones de replicación elegidas, pueden devolver datos modificados al publicador.

La fuente de la transacción replicada es llamada Base de Datos primaria, el destino de una transacción replicada es llamada Base de Datos replicada. Como se muestra en la **Fig.3.3**, escenario de replicación simple; la primera BD publica las transacciones replicadas, y la BD replicada las suscribe.

En este entorno de replicación, la BD primaria es el origen de los cambios que se replican a la BD destino. Si el mantenimiento de la BD destino se realiza únicamente para los informes y consultas, puede ser un subconjunto de la BD de producción, que contenga únicamente las tablas necesarias para los informes y las consultas, en lugar de copias de todas las tablas de producción.



En la **Fig.3.4**, se muestra la replicación bidireccional, en la cual una sola BD actúa como ambos, una BD primaria y una BD replicada. La replicación bidireccional requiere de un método de filtrado de las transacciones replicadas para prevenir la réplica circular a la fuente de BD original primaria.



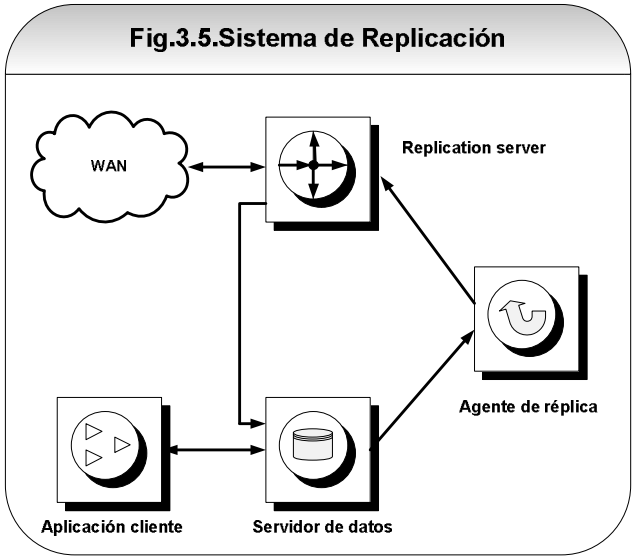
Cuando la alta disponibilidad es el objetivo primordial de la replicación de datos, la replicación bidireccional es la topología habitual. Esta configuración prevé que la actividad de actualización se realice en el sistema primario durante las operaciones normales y que, a continuación, se replique al sistema secundario.

El sistema secundario se configura para replicar los mismos datos a la inversa del sistema primario, por si la actividad del usuario debiera pasar a éste en caso de que se produjera un fallo del sistema principal.

Si eso se produce, la instancia de destino esta abierta y disponible, lista para permitir a las empresas continuar con una interrupción mínima, en lugar de necesitar un proceso largo de apertura y recuperación de la BD.

Replication Server tiene una arquitectura abierta que permite construir un sistema de replicación a partir de sistemas y aplicaciones ya existentes y agregarle cómo la organización crece y cambia.

La siguiente figura **Fig.3.5** es una descripción simplificada de un sitio del sistema de réplica basada en una WAN, un sistema distribuido de BD que usa Replication Server. En seguida se describe cada componente.



Servidor ID

El Servidor ID es un Replication Server que registra todos los Replication Servers y BD en el sistema de replicación. El ID Server funciona si:

- El Replication Server está instalado
- El encaminamiento (route) está creado
- La conexión de la BD está creada o abajo

El ID Server es el primer Replication Server que comienza cuando se instala un sistema de replicación.

El ID Server tiene que tener un nombre de conexión para los Replication Servers cuando estos se conecten al ID Server. El nombre de conexión está registrado en los archivos de configuración de todos los Replication Servers en el sistema de replicación por la configuración del programa rs_init.

Dominio del sistema de replicación

Se refiere a todos los componentes del sistema de replicación que utiliza el mismo ID Server. Se pueden instalar múltiples dominios del sistema de replicación, con las siguientes restricciones:

- Los Replication Server ubicados en diferentes dominios no pueden intercambiar datos entre ellos. Cada dominio tiene que ser tratado como un sistema de replicación separado sin comunicación entre ellos. No se puede crear una ruta entre Replication Servers en diferentes dominios.
- Una BD puede ser manejada por un solo Replication Server en un dominio. Cualquier BD está en uno y solamente en un dominio de servidor ID. Esto significa que no se pueden crear múltiples conexiones a la misma BD de diferentes dominios.

Replication Server[*29]

Un Replication Server en cada sitio coordina las actividades de replicación de datos para el servidor de datos local e intercambia datos con los Replication Servers en otros sitios.

Un Replication Server:

- ✓ Recibe transacciones de datos de la BD primaria o fuente vía el Agente de Réplica y los distribuye a los sitios que requieren esos datos.
- ✓ Recibe transacciones de otros Replication Servers y las aplica a la BD local.

Las tablas del sistema del Replication Server guardan la información necesaria para lograr estas tareas. Las tablas del sistema incluyen descripciones de los datos replicados y replicación de objetos, por ejemplo, definiciones de replicación y suscripciones, expedientes de seguridad para los usuarios del Replication Server, información del encaminamiento (route) para otros sitios, métodos de acceso para la BD local, y otra información administrativa.

Las tablas del sistema del Replication Server son almacenadas en una BD dentro del SMBD llamada Base de Datos del Sistema de Replication Server(RSSD). Un RSSD es asignado a cada Replication Server. Un servidor de datos con un RSSD puede también almacenar aplicaciones de BD.

[*29] Recopilación de información del libro [9] p. 300-301

Ambiente de Replicación

Un ambiente de replicación consiste de un sistema de servidores que participan en la replicación. Esto incluye los Servidores de Datos, el Agentes de Replicación, Replication Server. Un Ambiente de Replicación no tiene que incluir el dominio del sistema de replicación.

Manejador de Réplica

El Manejador de Réplica (RM) es instalado como un plug-in en el Sybase Central. RM es una utilidad de administración para conversiones, manejo y monitoreo de Ambientes de Replicación.

Servicios de Monitoreo de Replicación

El servicio de monitoreo de replicación (RMS) actúa como capa intermedia en una solución de 3 capas para un ambiente de replicación. RMS monitorea el estado de los servidores y de los componentes en el ambiente de replicación y proporciona información para localizar el problema e indica una posible solución.

Servidores de Datos

El servidor de datos maneja BD que contienen datos primarios o replicados. Los Clientes los utilizan para almacenar y recuperar datos y procesar preguntas y transacciones. Replication Server mantiene replicados los datos en los servidores de datos conectándose como un usuario de la BD.

Agente de Réplica

El Agente de Réplica transfiere la información del registro de la transacción, el cual representa cambios hechos en la BD primaria, de un Servidor de Datos a un Replication Server para la distribución a otras BD. El Agente de Réplica es RepAgent, el cual es una forma de dividir un programa en dos o más tareas que corren simultáneamente (thread).

El Agente de Réplica lee el registro de la transacción de la BD y transfiere los archivos para las tablas y procedimientos almacenados que serán replicados en el Replication Server que maneja la BD. El Replication Server reconstruye la transacción y la reenvía a los sitios que solicitan los datos.

Un Agente de Réplica es necesario para cada BD que contenga datos primarios o para cada BD donde los procedimientos almacenados replicados son ejecutados. Una BD que contiene únicamente copias de datos replicados y no tiene procedimientos almacenados replicados no requiere de un agente de replicación RepAgent.

Aplicaciones Cliente[*30]

Una Aplicación Cliente es un usuario del programa que accesa a un Servidor de Datos. Las aplicaciones pueden ser programas creados con Open Client/Server, Embedded SQL, Power Builder, o con cualquier herramienta de desarrollo compatible con Sybase Client/Server Interfaces (C/SI).

Las Aplicaciones Cliente actualizan únicamente los datos primarios. El Replication Server distribuye los cambios a otros sitios. Las Aplicaciones Cliente que no modifican datos no necesitan

[*30] Información proveniente del libro [9] p. 302

distinguir entre datos primarios y datos replicados. Las aplicaciones cliente pueden actualizar datos primarios usando alguno de los siguientes métodos:

- Conectándose directamente al servidor de la base de datos primaria cuando la actualización de operaciones sea requerida.
- Utilizando procedimientos de almacenamiento asíncrono. Cuando un cliente llama un procedimiento o almacenado asíncrono, el procedimiento almacenado actualiza los datos en el sitio primario y Replication Server distribuirá los cambios.

3.2.5 Tabla de replicación

Las transacciones replicadas son publicadas por una tabla, cuando las transacciones afectan los contenidos de una tabla publicada en la BD primaria, los datos son grabados por una distribución subsecuente a la replicación de la BD.

Una BD replicada puede ser un subconjunto de la BD primaria, con algunas, pero no todas las tablas que se encuentran en ella. Por lo tanto, no todas las tablas en la BD primaria tienen que ser replicadas.

Para recibir transacciones replicadas, la BD replicada tiene que suscribir a la tabla publicada en la BD primaria, y tiene que identificar la suscripción de la tabla de la BD replicada. Las transacciones replicadas que provienen de la BD primaria son distribuidas a las tablas suscriptoras en la BD replicada.

3.2.6 Réplica de procedimiento almacenado

Un procedimiento almacenado es un programa o procedimiento el cual es almacenado físicamente en una BD. La ventaja de un procedimiento almacenado es que al ser ejecutado, en respuesta a una petición de usuario, es ejecutado directamente en el motor de la BD, el cual usualmente corre en un servidor separado. Como tal, posee acceso directo a los datos que necesita manipular y solo necesita enviar sus resultados de regreso al usuario, deshaciéndose de la sobrecarga resultante de comunicar grandes cantidades de datos salientes y entrantes.

Usos típicos para procedimientos almacenados incluyen la validación de datos siendo integrados a la estructura de base de datos (los procedimientos almacenados utilizados para este propósito a menudo son llamados detonadores), o encapsular un proceso grande y complejo. El último ejemplo generalmente ejecutará más rápido como un procedimiento almacenado que de haber sido implementado como, por ejemplo, un programa corriendo en el sistema cliente y comunicándose con la base de datos mediante el envío de consultas SQL y recibiendo sus resultados.

Los procedimientos pueden ser ventajosos: Cuando una base de datos es manipulada desde muchos programas externos. Al incluir la lógica de la aplicación en la base de datos utilizando procedimientos almacenados, la necesidad de embeber la misma lógica en todos los programas que acceden a los datos es reducida. Esto puede simplificar la creación y, particularmente, el mantenimiento de los programas involucrados.

Podemos ver un claro ejemplo de estos procedimientos cuando requerimos realizar una misma operación en un servidor dentro de algunas o todas las bases de datos y a la vez dentro de todas o algunas de las tablas de las bases de datos del mismo. Para ello podemos utilizar a los procedimientos almacenados autocreables que es una forma de generar ciclos redundantes a través de los procedimientos almacenados.

Además de la replicación de transacciones, otra forma de mantener consistentes datos sincronizados, es replicar la invocación de los procedimientos almacenados que cambian los datos. La replicación de una invocación de un procedimiento almacenado puede, algunas veces, ser más eficiente que la replicación individual de las transacciones.

Cuando un procedimiento almacenado es publicado, el sistema de replicación tiene que identificar el procedimiento y grabar los valores del parámetro de entrada que son especificados cuando el procedimiento es invocado. El sistema entonces debe distribuir aquella invocación de procedimiento a cualquier base de datos replicada suscrita.

La replicación de un procedimiento almacenado coloca un requerimiento especial sobre un sistema de replicación. Cuando un procedimiento publicado genera una transacción sobre una tabla publicada, el sistema de replicación debe ser capaz de reconocer la operación generada por el procedimiento publicado, y replicar únicamente la invocación del procedimiento y no la transacción producida por el.

3.2.7 Réplica de transacción

La réplica de transacción asegura integridad y consistencia transaccional entre las bases de datos. Todas las operaciones que cambian los datos que son replicados, son consideradas como transacciones, aun cuando ellos no pudieran corresponder a una transacción real en la base de datos primaria.

Por ejemplo, si una transacción actual cambia ambas tablas, publicadas y no publicadas en la BD primaria, solo las transacciones sobre la tabla publicada son replicadas. Las operaciones sobre las tablas no publicadas no se replican, pero la consistencia transaccional es mantenida si la BD que replica contiene únicamente tablas que corresponden a tablas publicadas en la BD primaria.

Aún cuando la replicación de una transacción sea solamente un juego de operaciones que cambian datos, esas operaciones son agrupadas en una pequeña colección, y cada colección representa los resultados de una transacción completada (commit) en la BD primaria. Únicamente las operaciones de transacción completadas deberían ser replicadas; las operaciones de transacción de roll back no deberían ser replicadas.

Las invocaciones de procedimientos almacenados son consideradas parte de una transacción, así como las operaciones que cambian los datos sobre la tabla. Las invocaciones de procedimiento no son necesariamente transacciones en si mismas.

3.2.8 Entrega garantizada

En un sistema de replicación, entrega garantizada significa que todas las transacciones o invocaciones de procedimientos publicados por una BD primaria son entregadas y recibidas por la BD replicada que suscribe a pesar del hardware, software o problemas en la red que interfieran con la replicación.

Para proveer la entrega garantizada, el SMBD usa dos mecanismos:

- Una cola estable, la cual graba las transacciones replicadas en una forma no volátil (en disco), hasta que la suscripción en la base de datos replicada confirma que lo ha recibido.
- Un localizador, identifica la última transacción publicada que fue recibida y almacenada satisfactoriamente en la cola, y la última transacción publicada que fue recibida satisfactoriamente por la suscripción de la BD replicada.

3.3 SYBASE[20]

A principios de la década de 1990, los analistas predijeron que Sybase desbancaría a Oracle y se convertiría en el nuevo rey de la colina. Fue precursor de tecnologías importantes, como los procedimientos almacenados, triggers, replicación de datos, servidores multihilados, servidores abiertos, BLOB (Binary Large OBjects, grandes objetos binarios) y middleware de SQL.

Como los demás, Sybase tiene una estrategia del lado servidor que se inclina por Java. El elemento principal de la estrategia es el servidor de aplicaciones Jaguar Component Transaction Server.

3.3.1 Sybase ASE

Sybase ASE es un servidor de bases de datos relacionales, Software y Administración de datos. El producto Adaptive Server Enterprise incluye:

- Adaptive Server
- Backup Server
- Monitor Server
- XP Server
- Web Services
- Utilidades de Adaptive Server
- Archivos de secuencias de comandos y de configuración

Adaptive Server

Es un servidor de base de datos relacional. Sybase Central (Plug-in de Adaptive Server) es un sistema marco común para la administración de servidores. Permite administrar las instalaciones de Adaptive Server mediante la herramienta gráfica de administración Sybase Central.

Backup Server

Backup Server es una aplicación basada en Open Server (Sistema operativo servidor UNIX escalable, confiable y flexible, copia de seguridad y recuperación de datos, componentes de cliente y servidor de correo electrónico, entre otras características) que gestiona todas las operaciones de copia de seguridad y restauración de bases de datos del servidor Adaptive Server. Permite:

- Utilizar un número limitado de dispositivos de backup que se denomina de forma paralela para el backup o carga de un diario de transacciones o bases de datos.
- Realizar backup comprimidos y cargas en el disco local.
- Realizar un backup en varias cintas o transferir varios backup a una única cinta.
- Operaciones bidireccionales de backup y carga en la red con un dispositivo situado en otro equipo.
- Determinar automáticamente con comandos del sistema operativo las características de los dispositivos de cinta para una operación de backup.
- Admite la especificación de sintaxis de comandos de backup y carga para la asignación de nombres de volúmenes, el control de desmontar y cargar, la densidad de cinta, el tamaño de bloques, la capacidad de cinta, los días de retención, la inicialización, la asignación de información de archivos o cabecera.

Se instala Backup Server si se tiene previsto realizar una copia de seguridad y restaurar las bases de datos.

Sybase Monitor Server

Aplicación Open Server que obtiene estadísticas de rendimiento sobre el servidor Adaptive Server y que las pone a disposición de las aplicaciones cliente Monitor Server, incluye:

- Monitor Server, aplicación Open Server que obtiene estadísticas de rendimiento del servidor Adaptive Server que se pueden consultar con los monitores Monitor Historical Server y las aplicaciones creadas con la biblioteca Monitor Client-Library.
- Monitor Client-Library es una interfaz de programación que proporciona acceso a los datos de rendimiento de Adaptive Server.
- Monitor Historical Server, aplicación Open Server que obtiene estadísticas de rendimiento de varios servidores Adaptive Server a través de servidores Monitor Server y que registra los datos en las ubicaciones de archivo especificadas.

XP Server

Aplicación Open Server que gestiona y ejecuta procedimientos almacenados extendidos (ESP) desde el servidor Adaptive Server. Los procedimientos ESP proporcionan un método de llamada para las funciones de lenguaje de procedimientos desde Adaptive Server.

Para configurar XP Server y para conectar XP Server y Adaptive Server mediante el archivo de interfaces se hace uso de la utilidad srvbuild (utilizado para la creación de un servidor funcional, Interfaz gráfica de usuario X-Windows/Motif).

Web Services

Un servicio Web es un uso autónomo, modular que se puede alcanzar a través de una conexión de red. Usando un servicio Web, el usuario final negocia el funcionamiento para la interoperabilidad creciente, hecho por el Simple Object Acces Protocol (SOAP), Web Services Description Language (WSDL), HTML y Extensible Markup Language (XML).

Sin importar el lenguaje de programación en el cual se ha puesto en ejecución, un servicio Web se puede alcanzar de diversas plataformas y sistemas operativos, de esta manera dar disponibilidad a la capacidad para que las diversas aplicaciones compartan datos.

ASE Web Services se conforma de dos componentes, un productor de servicios Web y un consumidor de servicios Web. Ambos componentes corren independientemente de ASE y son habilitados por la misma licencia. El productor de servicios Web habilita las aplicaciones del cliente para accesar al SQL y a los procedimientos almacenados en ASE utilizando SOAP. El consumidor de servicios Web habilita ASE para accesar a un servicio Web de otras aplicaciones.

Utilidades de Adaptive Server

Proporciona un controlador (jConnect) de conectividad de bases de datos Java (JDBC) que funciona en máquinas virtuales (VM) Sun y Microsoft. Proporciona compatibilidad para límites extendidos de Adaptive Server al solicitar compatibilidad para tablas más anchas.

Utilidad de Java. Cascada Gateway, pasarela que actúa como Proxy para proporcionar una ruta al servidor de bases de datos si se ejecuta en un servidor distinto al servidor Web. Jisql, editor gráfico Transact-SQL desarrollado en Java y que sustituye a SQL Advantage. Ribo, utilidad que captura, convierte y muestra los flujos del protocolo TDS (Tabular Data Stream, Flujo de datos en tablas) entre un cliente TDS y un servidor TDS.

Controlador ODBC permite a las aplicaciones cliente Windows NT tener acceso a los datos del servidor Adaptive Server.

Open Client contiene bibliotecas y utilidades para desarrollar aplicaciones Open Client.

Open Client

Sirve para desarrollar y distribuir aplicaciones basadas en el lenguaje C que tienen acceso a los datos del servidor Adaptive Server. Incluye bcp, utilidad para transferir datos entre archivos y bases de datos; isql, utilidad para conectar al servidor Adaptive Server para las consultas de SQL; MonitorClient-Library, interfaz de programación de aplicaciones que proporciona acceso a los datos de rendimiento de Adaptive Server; SQL Remote, permite replicaciones bidireccionales entre un servidor de base de datos y varias bases de datos de equipos portátiles mediante el correo electrónico o conexiones de acceso telefónico a redes; arquitectura física, herramienta para crear modelos de datos, incluido el diseño, la generación, el mantenimiento, la ingeniería inversa de bases de datos y la documentación de arquitectura de bases de datos.

3.3.2 Sybase Replication Server

Sybase Replication Server proporciona una replicación bidireccional, heterogénea y sincronizada a través de la empresa, cliente/servidor y sistemas móviles, entrega datos operacionales a través de estructura de datos en tiempo real, mantiene datos replicados en múltiples BD mientras asegura la integridad y consistencia de los datos.

- ✓ Se asegura de que los datos estén continuamente disponibles
- ✓ Asegura la pronta recuperación de un desastre
- ✓ Soporte oportuno y divulgación cuidadosa de los datos
- ✓ Soporte de plataformas heterogéneas de la base de datos - Sybase ASE, Oracle, IBM DB2 y servidor de Microsoft SQL
- ✓ Proporciona clientes que utilizan bases de datos en el sistema de la réplica con acceso local de los datos, de esta manera reduce la carga en la red y en los sistemas informáticos centralizados.

El Replication Command Lenguaje (RCL) permite modificar funciones de la réplica para requisitos particulares, supervisar y mantener el sistema. Por ejemplo, puedes solicitar subconjuntos de los datos para la réplica en la tabla, la fila de datos o el nivel de la columna. Esta característica te permite replicar solamente los datos que te sean necesarios.

3.3.2.1 Distribución de datos con Replication Server

Replication Server funciona para distribuir datos sobre una red de la siguiente manera:

- ➢ Provee aplicaciones de uso y sistemas administradores con un modelo flexible de publicación y suscripción para datos específicos y procedimientos almacenados que son replicados.
- ➢ Manejando transacciones replicadas mientras conserva la integridad de la transacción a través de la red.

Porque Replication Server replica transacciones (cambios incrementales en lugar de copias completas de datos) e invocaciones de procedimientos almacenados, no los procedimientos almacenados en sí mismos, proporciona un ambiente distribuido de alto rendimiento de los datos mientras mantiene integridad en los datos.

3.3.2.2 Modelo de publicación y suscripción

En un sistema de Replication Server funcionando, las transacciones ocurren en la base de datos fuente y son detectadas por Replication Agent (antes mencionado), y transferido al Replication Server local, el cual distribuye la información através de la LAN y WANs a los Replication Server destino. Estos Replication Servers actualiza los objetivos de la base de datos deacuerdo a los requerimientos del cliente remoto. Si la red o los componentes del sistema fallan, el dato en el proceso de ser entregado es temporalmente almacenado en colas. Cuando el componente que ha fallado retorna a la operación, el sistema de replicación resincroniza las copias de los datos y el resumen de la replicación normal.

El dato a replicar es la fuente que Replication Server replica en otra BD. Primero se crea una definición de la réplica para señalar la localización de los datos primarios. La definición de replicación describe la estructura de la tabla y los nombres de las BD que contiene la copia primaria de la tabla. Para un manejo más fácil, se pueden recolectar definiciones de la réplica en las publicaciones.

Una suscripción se asemeja a un select en SQL, puede incluir cláusulas que especifiquen qué campos de una tabla puedes replicar en una BD local, permitiendo que replique únicamente los datos necesarios.

Se pueden tener múltiples definiciones de replicación para una tabla primaria. Las tablas replicadas pueden suscribir diferentes definiciones de replicación para obtener diferentes vistas de los datos.

Una vez que se hayan creado las suscripciones a las definiciones de replicación o publicaciones, Replication Server replica las transacciones a la BD con suscripciones para los datos.

3.3.2.3 Transacciones replicadas

Replication Server permite replicar invocaciones de procedimientos almacenados del Adaptive Server de manera asíncrona entre BD. Este método puede mejorar la réplica de los datos dado que encapsula muchos cambios en una sola función replicada. Porque no están asociados con una tabla donde existan definiciones de replicación, las funciones de replicación pueden ejecutar procedimientos almacenados que pueden o no modificar datos directamente.

Se pueden replicar invocaciones de procedimientos almacenados de una BD primaria a una BD replicada, o de una BD replicada a una BD primaria.

Con funciones replicadas, se puede ejecutar un procedimiento almacenado en otra BD. Una función replicada permite replicar la ejecución de un procedimiento almacenado del Adaptive Server a un sitio de suscripción y mejorar el funcionamiento replicando únicamente el nombre y los parámetros del procedimiento almacenado más que los cambios actuales.

Como las tablas, los procedimientos almacenados replicados quizá tengan definiciones de replicación, las cuales son llamadas definiciones de replicación de función, y suscripciones. Cuando un procedimiento almacenado replicado es ejecutado, el Replcation Server pasa su nombre y parámetros de ejecución a un sitio de suscripción, donde el procedimiento almacenado se ejecuta.

Se crean definiciones de replicación de función en el sitio de datos primario. Replication Server soporta funciones aplicadas y funciones de petición:

Una función aplicada es replicada de una BD primaria a una BD replicada. Se crean suscripciones en el sitio replicado para la definición de replicación de función y marca el procedimiento almacenado para la replicación en la BD primaria.

Una función de petición es replicada de una BD primaria. No hay suscripción para una función de petición. Se marca el procedimiento almacenado para la replicación en la BD replicada.

CAPÍTULO IV PROPUESTA DEL SISTEMA DE ALTA DISPONIBILIDAD

Este capítulo presenta la propuesta del sistema de alta disponibilidad en bases de datos para el departamento de servidores de la DGSCA, respuesta a las necesidades del departamento para una de las aplicaciones que administra, "Tiendas electrónicas UNAM".

Para dar entrada a la estructura del proyecto, primero hay que visualizar la estructura organizacional de la Institución a la que se le ha propuesto el desarrollo del proyecto y el departamento encargado de su implementación.

4.1 LA DGSCA

La Dirección General de Servicios de Cómputo Académico de la UNAM es la entidad universitaria encargada de la operación de los sistemas centrales de cómputo académico y de las telecomunicaciones de la institución; su esfuerzo más amplio es la capacitación en tecnología de la información, de prospección e innovación y de asimilación de estas tecnologías en beneficio de la Universidad y de la sociedad en general.

La capacidad de respuesta siempre es adaptada constantemente a la evolución de la tecnología. La actividad se integra con las actividades de escuelas y facultades de la UNAM, con sus institutos y centros de investigación y con muchas otras de sus entidades. El interés es responder continuamente a sus demandas de servicios y de formación de recursos humanos. De igual manera, la DGSCA trabaja con el sector público y con empresas privadas, dentro de un marco de exigencia y creatividad.

Todos los servicios de supercómputo, Internet, servidores y sistemas, están preparados para trabajar con grandes volúmenes de cómputo numérico y visual, con tasas altas de transacciones y transmisión de datos, multimedia con elevados niveles de integración de información, en un esquema regido por la calidad y seguridad.

En todas las tareas se generan nuevos aprendizajes para las personas y para la institución. Se formulan las actividades para que contribuyan a la formación integral de los miembros de la organización y se exige que las nuevas tareas que se emprenden impliquen retos que redunden en nuevos aprendizajes.

Se impulsan los usos de las tecnologías de la información que permiten más eficazmente la comunicación e interacción entre personas y organizaciones, así como entre sistemas. Extiende su capacidad de prueba de sistemas como un servicio a todas las organizaciones con las que colabora.

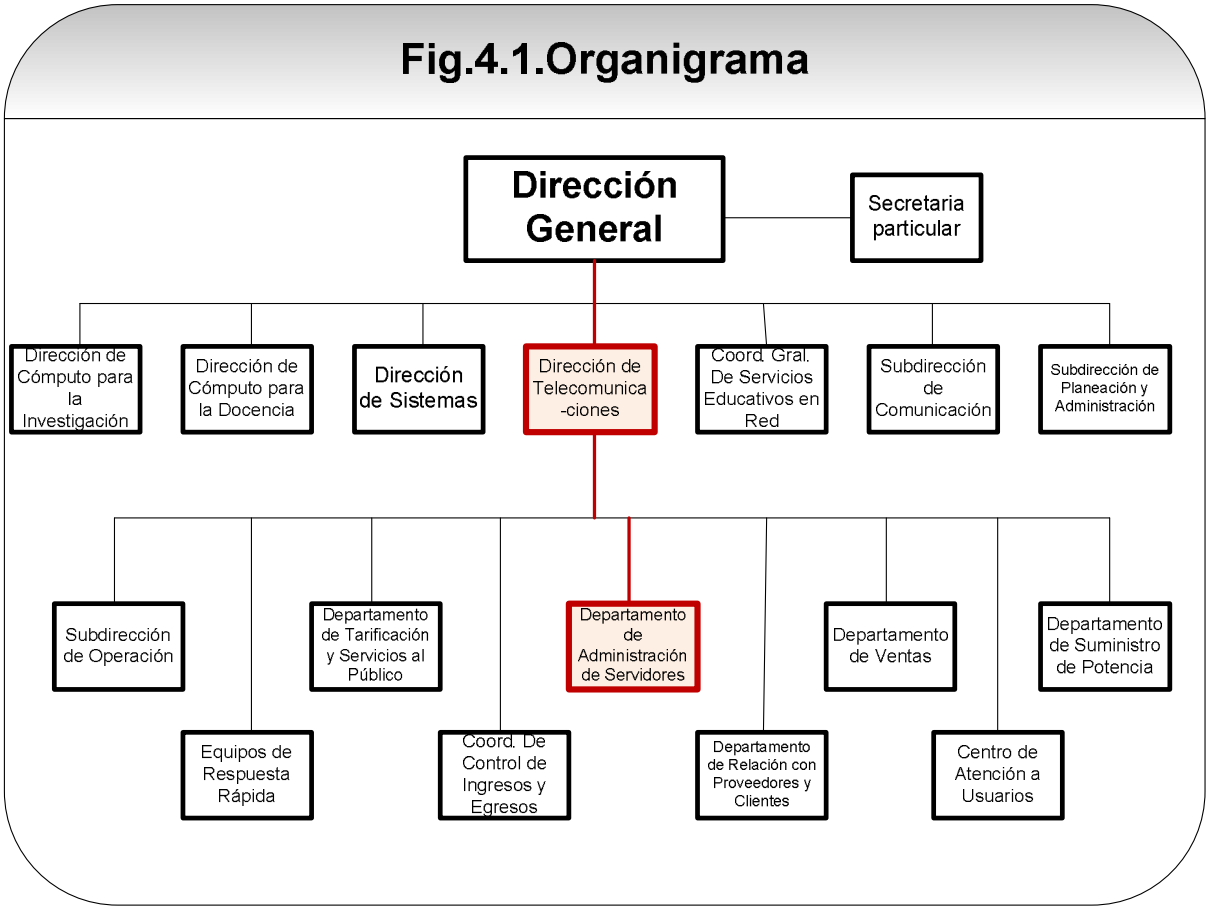
Se realiza investigación, se impulsan nuevos desarrollos en métodos numéricos, algoritmos y computación visual. Se exploran las innovaciones tecnológicas más recientes y se desarrollan nuevos sistemas.

La DGSCA está conformada por personas que conciben el servicio centrado en el usuario. Existe obligación de proveer servicios a la UNAM al más bajo costo posible y con una capacidad consistente.

En los grupos de Cómputo Infantil, Multimedia, Productos Interactivos para la Docencia, Servicios de Red, Administración de Servidores, Seguridad en Cómputo, Sistemas, Tecnología para la Educación a Distancia y muchos más, la actividad realizada se basa en investigaciones propias y en la innovación.

La Institución educa y capacita, los cursos, diplomados, programas de becas, y colaboraciones interinstitucionales tienen como objeto la educación de las personas en tecnologías de información, en su integración a las tareas de las organizaciones modernas.

4.1.1 Organigrama



4.1.2 Dirección de Telecomunicaciones Digitales

La Dirección de Telecomunicaciones de la DGSCA apoya funciones universitarias sustantivas en áreas de investigación, docencia y difusión cultural, además brinda soporte a las actividades administrativas de toda dependencia de la UNAM. Todo esto a través de una moderna y robusta infraestructura de cómputo y telecomunicaciones que le permite integrar su propia red de voz, datos y videoconferencia.

Misión

Proporcionar a la UNAM la infraestructura en telecomunicaciones que le permita desarrollar sus actividades sustanciales en proporción a su desarrollo y crecimiento.

Visión

Ser la Institución educativa líder en aplicaciones de telecomunicaciones consideradas de vanguardia.

4.1.3 El departamento de Administración de Servidores

El departamento de Administración de Servidores permite que se reduzca la complejidad, el tiempo de inactividad y los costos de los servicios de Web, Bases de Datos, correo, antiespam entre otros, monitoreando, implementando proyectos y automatizando sus servicios con tecnología reciente, mientras se cumplen con las políticas y los estándares de la Institución. Las soluciones que el departamento busca, ofrecen administración centralizada e inventarios de entornos con diversos sistemas operativos y hardware, implementación y migración de sistemas operativos y servidores completos, administración de software y parches. Sus principales actividades para con los usuarios son:

Asesorías
El departamento de Administración de Servidores brinda asesoría técnica en cuentas de correo.

Bases de datos
Asignación de recursos para la creación de bases de datos.
Solicitud de respaldo de las bases de datos.

Correo
Creación y borrado de cuentas de correo administradas por el departamento, si así se requiere.
Cambio de password's.
Administración de cuentas de correo de los sitios que tienen Hospedaje Web.
Desbloqueo de cuentas de correo.
Atiende fallas de operatividad de login o contraseña en el correo electrónico.
Realiza recuperación de correos en caso muy necesario.

Internet
Asignación gratuita de acceso a Internet vía módem para académicos, funcionarios y empleados.
Brinda ayuda para configurar el acceso a Internet vía módem.

Servidores
Acceso a la sala de la DGSCA (alojo de servidor).
Alojo de sitios Web solicitado por dependencias de la UNAM.
Alojo en servidor solicitado por dependencias de la UNAM.

Otros
Actualización de datos de los administradores para servicio DNS.
Difusión de actividades relacionadas con la UNAM a través del Web o correo.
Falla de administración en un servidor DNS.

4.2 TIENDAS ELECTRÓNICAS UNAM

Tiendas Electrónicas UNAM aprovecha las muchas ventajas de Internet y el comercio electrónico como vehículo para dar un mejor servicio a los compradores. Su dirección física se encuentra en la Dirección General de Servicios de Cómputo Académico circuito exterior s/n.

La dirección electrónica: www.etienda.unam.mx, ver APÉNDICE 1.

La aplicación se encuentra programada en PHP.

Como recordaremos, en el tema de comercio electrónico, una de las prioridades es la seguridad, tiendas electrónicas UNAM, cuenta con un certificado digital expedido por Verisign que asegura que se realicen transacciones seguras y protegidas; con éste certificado se puede confiar en que se cuenta con la tecnología de encriptación SSL de 128 bits. Cada transacción entre el navegador del cliente y el servidor está segura y protegida por este certificado.

Tiendas Electrónicas UNAM se conforma de las siguientes dependencias de las cuales, cada una de ellas conforma una tienda electrónica:

- ✓ Dirección General de Servicios de Cómputo Académico
- ✓ Instituto de Investigaciones Jurídicas
- ✓ Instituto de Investigaciones Económicas
- ✓ Instituto de Investigaciones Sociales
- ✓ Dirección General de Divulgación de la Ciencia
- ✓ Facultad de Ingeniería
- ✓ Dirección General de Estudios de Postgrado
- ✓ Facultad de Medicina
- ✓ Dirección General de Literatura
- ✓ Centro Universitario de Investigaciones Bibliotecnológicas
- ✓ Instituto de Investigaciones Filosóficas
- ✓ Programa Universitario de Estudios de Género
- ✓ Instituto en Investigaciones de Matemáticas Aplicadas y en Sistemas
- ✓ Facultad de Psicología
- ✓ Instituto de Biología
- ✓ Facultad de Estudios Superiores Cuautitlán
- ✓ Facultad de Contaduría y Administración
- ✓ Centro de Investigaciones Interdisciplinarias en Ciencias y Humanidades
- ✓ Instituto de Investigaciones Filológicas
- ✓ Instituto de Investigaciones sobre la Universidad y la Educación
- ✓ Facultad de Economía
- ✓ Dirección General de Bibliotecas
- ✓ Centro de Investigaciones sobre América Latina y el Caribe
- ✓ Oficina del Abogado General
- ✓ Facultad de Ciencias Políticas y Sociales
- ✓ Centro Regional de Investigaciones Multidisciplinarias

El sitio Web se encuentra hospedado en el servidor AURA, brinda servicio mediante un servidor HTTP Apache 1.3.34, software libre de código abierto que implementa el protocolo HTTP. Su Base

de Datos se encuentra en el Sistema Manejador de Base de Datos Sybase ASE-12.5.

El portal de Tiendas UNAM es visitado por bastantes países alrededor del mundo; debido a esto Tiendas UNAM cuenta con una tarifa especial de entrega dependiendo el lugar de dónde se solicita el producto, brinda servicio a:

- Toda la República Mexicana
- Estados Unidos y Canadá
- Asia, Pacífico y Medio Oriente
- Europa
- Centro, Sudamérica y el Caribe
- África

4.2.1 GRAFICAS DE ACCESO CON GEOLIZER

Geolizer es un pequeño programa hecho en C el cual nos permite generar reportes de alguna página Web. Gracias a esos reportes, podemos observar el número de personas que han entrado en la página. Este programa no sólo nos da los reportes cuantitativos, sino que también nos da reportes gráficos, lo que lo hace más entendible y sencillo al observar las estadísticas.

Entre otras cosas Geolizer es sumamente útil para saber qué archivos son los que poseen más número de descargas en la página. Cuando poseemos una página Web y cualquier persona de la Internet accede a ésta eso siempre queda registrado en una bitácora, que comúnmente se le denomina "logs". Una vez que esas bitácoras del sistema quedan registradas en un archivo llamado access_log, el programa Geolizer los analiza y genera gráficas en formato HTML para que puedan ser observadas desde el navegador.

Lo que hace es leer el formato del archivo de bitácoras, todos esos datos los analiza y genera un archivo html accesible desde la Web. Este archivo html, tiene reportes con gráficas de todo lo que ha sido el número de visitas, tráfico, archivos ofrecidos, el tipo de navegadores Web que mas visita nuestra página, etc.

Veamos la siguiente información arrojada por el programa Geolizer con la finalidad de tener un panorama más amplio de la importancia y movimiento de la aplicación “Tiendas electrónicas UNAM” en la Web.

La siguiente información es resultado del procesamiento de los logs de la aplicación Web de tiendas electrónicas UNAM de una muestra correspondiente al mes de Octubre:

Gráficas de Octubre

Fig.4.2.Datos de Octubre

Estadisticas del sitio tiendadgc www.etienda.unam.mx

Periodo resumido: Octubre 2007
Generado el 01-Nov-2007 20:04 CST
GEO-106FREE 20070201 Build 1 Copyright (c) 2007 MaxMind LLC All Rights Reserved

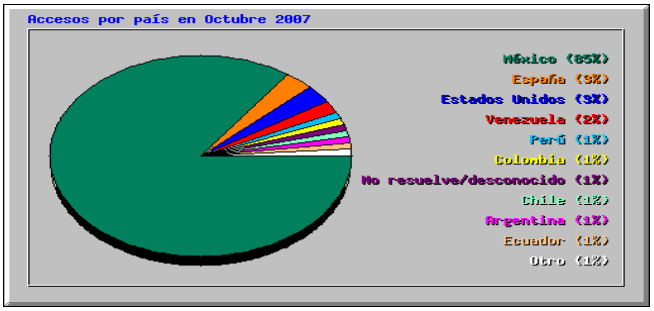
[Estadísticas diarias] [Estadísticas por horas] [URLs] [Entrada] [Salida] [Clientes] [Enlaces origen] [Búsqueda] [Usuarios] [Navegadores] [Países]

Estadísticas mensuales de Octubre 2007		
Total Accesos	1292684	
Total Archivos	983697	
Total Páginas	189725	
Total Visitas	26316	
Total KBytes	9.59 GB	
Total Clientes	16163	
Total URLs	3021	
Total Enlaces origen	1690	
Total Usuarios Únicos	7	
Total Navegadores	2505	
	Media	Max
Accesos por Hora	1737	9189
Accesos por Día	41699	57911
Archivos por Día	31732	43344
Páginas por Día	6120	9128
Visitas por Día	848	1085
KBytes por Día	316.90 MB	451.05 MB

En la **Fig.4.2.** se puede visualizar el número total de peticiones hechas al servidor en el transcurso del mes, por holgura se toman los números máximos; es decir, los resultados anteriores muestra 9189 accesos máximos por hora, es decir, aproximadamente 153 accesos por minuto. 983697 archivos es el total de peticiones devueltas a los usuarios en el transcurso del mes. Con esta información podemos visualizar un poco el movimiento que se genera con esta aplicación.

Visualicemos ahora los accesos por país:

Fig.4.3.Accesos por país de Octubre



La **Fig.4.3.** nos muestra los accesos realizados por clientes en los distintos países del mundo, no únicamente por México. A pesar de los pocos accesos realizados por los demás países al sitio Web, la aplicación pretende extender y aumentar sus ventas tanto en el interior de la República Mexicana como en otros países.

Los datos anteriores nos proporcionan un panorama más amplio de Tiendas electrónicas UNAM, ésta es visitada por varios países alrededor del mundo, principalmente México, y tiene un número considerable de accesos promedio al día que realizan compras a través de este medio, hablamos de un negocio con cobertura amplia en toda la República Mexicana y países del mundo. Es un negocio que se encuentra en continuo crecimiento y requiere de alta disponibilidad para cumplir con sus objetivos.

4.3 PROPUESTA

4.3.1 Punto a tratar

Como ya observamos en el tema anterior "Tiendas electrónicas UNAM", la aplicación es visitada por distintos países alrededor del mundo y tiene un número considerable de accesos promedio al día que realizan compras a través de este medio, hablamos de un negocio con cobertura amplia en toda la República Mexicana y países del mundo; el objetivo es proporcionar un servicio ininterrumpido lo más próximo a las 24 horas del día, 7 días a la semana; recordemos que ninguna empresa cuenta con el 100% de disponibilidad debido a la complejidad de éste; la caída del servicio sin duda alguna afectaría el negocio.

De lado de los servidores, donde se brinda el hospedaje de la aplicación; aparte de brindar alta disponibilidad se propone el balanceo de carga. Esta propuesta se debe a que el actual servidor que aloja la página Web, también da alojo a 25 páginas más, páginas cuyos usuarios aumentan conforme pasa el tiempo y provocan una carga considerable en el servidor actual y por tanto lentitud de respuesta. Entonces se debe brindar el recurso necesario para administrar las solicitudes de un gran número de usuarios. De acuerdo con el capítulo anterior, realizando balanceo de carga estamos brindando al sistema escalabilidad debido al continuo crecimiento del número de usuarios y prontitud de respuesta a peticiones por parte de los clientes.

4.3.2 Otras opciones de solución

Existen diversos proyectos que se basan en la tecnología "clustering", es decir, un conjunto de computadoras que realizan determinados procesos y que en conjunto se muestran como un único sistema con capacidades superiores a las que podría brindar una sola computadora. A continuación mencionaré algunos proyectos que son los más conocidos, brindan este tipo de tecnología y hacen uso de software de código abierto; por tanto, podrían ser considerados para brindar solución a la problemática arriba descrita.

Piranha

Este proyecto brinda alta disponibilidad y balanceo de carga, su objetivo es similar al proyecto de UltraMonkey, sin embargo, utiliza distintas herramientas para llevar a cabo su objetivo. Por ejemplo, utiliza **nanny**, un demonio que monitorea los servidores y servicios en los servidores reales; **pulse**, demonio que se encarga de identificar que los servidores estén en funcionamiento. Cuenta con una herramienta de administración gráfica que administra el ambiente cluster. Sin embargo, también hace uso del Linux Virtual Server y la topología es idéntica a la que utiliza UltraMonkey. Piranha únicamente soporta NAT.

Keepalived

Es otra solución que implementa el proyecto Linux Virtual Server, brinda robustez por su topología, la topología es similar a la de UltraMonkey, sin embargo utiliza otro tipo de herramientas para llevar a cabo alta disponbiilidad; utiliza VRRPv2 para manipular a los directores y llevar a cabo el takeover.

UltraMoneky

Es la solución que hemos venido estudiando, también está basada en el proyecto Linux Virtual Server. Hace uso de herramientas confiables como heartbeat para monitorear los balanceadores; utiliza Linux-director como balanceador de carga. En su página oficial muestra de forma sencilla el funcionamiento y objetivos del proyecto UltraMonkey. La configuración de las herramientas y la red para la implementación de balanceo de carga y alta disponibilidad es relativamente sencilla.

Beowulf

Beowulf es un proyecto cuyo objetivo está enfocado en implementar la tecnología clustering con el objetivo de presentar el cluster como un único sistema. La idea consiste en que los procesos que son realizados por los servidores reales son manejados por un servidor llamado nodo maestro dando la impresión de que el clúster es un único sistema. En éste proyecto nos haría falta implementar alguna herramienta que nos brinde alta disponibilidad en el nodo maestro, porque en caso de un incorrecto funcionamiento del nodo maestro, el sistema definitivamente dejaría de funcionar.

Beowulf utiliza herramientas y bibliotecas usadas para presentar el cluster como único sistema. Las bibliotecas que utiliza para transferencia de mensajes entre los nodos pueden ser MPI (Message Passing Interface) o PVM (Prallel Virtual Machine), utiliza rsh y es necesario hacer parcheo del kernel.

OpenMosix

Es un proyecto cuyo funcionamiento es similar al proyecto Beowulf, de igual forma se tendría que buscar una solución para brindar alta disponibilidad al nodo maestro. Utiliza un conjunto de extensiones del kernel, brinda un herramientas que permiten presentar el sistema como único sistema (SCE, Scalable Cluster Environment), cuenta con licencia GPL (licencia publica general), reparte los procesos entre los servidores según su carga de forma automática, transparente y dinámica, es necesario parchar el kernel.

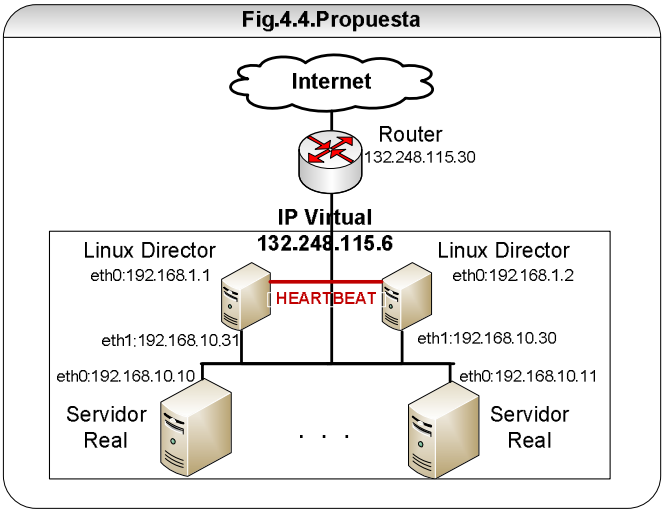
Al dar una rápida introducción del funcionamiento y herramientas que utilizan cada uno de los diversos proyectos que nos ayudan a cumplir con alta disponibilidad y balanceo de carga, por las características que los distinguen, los proyectos más viables que podrían servir para darle solución a la problemática son Piranha, UltraMonkey y Keepalived debido a que hacen uso del Linux Virtual Server; los otros dos proyectos, Beowulf y Openmosix se les necesitaría implementar la herramienta de alta disponibilidad en el nodo master. De éstas posibles opciones, me pareció viable utilizar el proyecto de UltraMonkey, las razones son las siguientes:

- Proporciona información suficiente y explícita acerca de la topología de alta disponibilidad y balanceo de carga y su configuración.
- La configuración del proyecto se muestra de una forma relativamente sencilla
- La investigación realizada en los foros de Internet acerca de la implementación del proyecto

de Ultramonkey es satisfactoria; existen algunas problemáticas a lo largo del desarrollo del proyecto, sin embargo, tienen solución. La confiabilidad que brinda el proyecto y su implementación cuenta con altas probabilidades de éxito.

4.3.3 Estructura de la propuesta

Para cumplir con el objetivo nos basaremos en la Topologia de "Alta Disponibilidad y Balanceo de Carga" **Fig.4.4.**



Contamos con dos máquinas que funcionan como servidores virtuales, Mopeia y Dhori, que se encuentran enfrente, Mopeia está configurada como master y es la máquina que recibirá las peticiones de los clientes, realizará balanceo de carga y las redirigirá a los servidores reales Aura o Azuki, servidores que contienen la aplicación y la Base de Datos, quienes en realidad procesarán la información y devolverán la información ya procesada para ser retornada al cliente.

La máquina Dhori se encontrará en "espera", esta máquina estará monitoreando a Mopeia mediante HeartBeat, en el momento en que Mopeia deje de funcionar Dhori tomará su lugar, logrando de ésta manera Alta Disponibilidad en la parte de enfrente.

Ambos servidores reales, Aura y Azuki, tienen repartida la carga de modo que se encontrarán procesando la información requerida por los clientes; en el momento en que uno de estos servidores falle, el otro servidor real estará dando respuesta a las peticiones. Debido a que es una aplicación que trabaja con bases de datos, ambos servidores deben tener la misma información; para lograrlo, los servidores reales realizarán replicación de datos entre ellos de forma síncrona mediante Sybase Replication Server.

Los detalles de red se verán en el subtema 4.3.6.Red

4.3.4 Hardware de las máquinas

Máquinas Intel Mopeia y Dhori

- Arquitectura x86
- Procesador Intel Pentium 4 de 3.4 Ghz
- Memoria RAM de 1 GB
- Disco duro de 160 GB

Máquina SunFire V20z Aura y Azuki

- Dos procesadores tipo AMD Opteron 250 de 64-bit
- Memoria caché con capacidad de 2 MB, 1 MB por procesador
- Memoria RAM de 4 GB tecnología DDR SDRAM – ECC con velocidad de 333 MHz
- Temperatura mínima para operar: 10 °C
- Temperatura máxima para operar: 35 °C

4.3.5 Software

Librerias básicas en la instalación del Sistema Operativo

- gcc
- gcc-java, soporte Java para gcc
- rcs, Revision Control System, manejo de herramientas
- automake15, herramienta GNU para la creación automática de Makefiles
- glibc-utils, utilidades de desarrollo de la librería C de GNU
- patchutils, conjunto de programas para la manipulación de parches

Se recomienda la instalación de todas las librerías incluidas en el disco de instalación del Sistema Operativo para evitar futuras solicitudes de estas.

Servidores virtuales Mopeia y Dhori

- Sistema Operativo Red Hat Enterprise Linux versión 4, kernel 2.6.9-5.
- La instalación de Ultramonkey incluye los siguientes rpm's:
  - arptables_jf-0.0.7-0.3E.i386.rpm
  - arptables-noarp-addr-0.99.2-1.rh.el.um.1.noarch.rpm
  - heartbeat-1.2.3.cvs.20050128-1.rh.el.um.1.i386.rpm
  - heartbeat-ldirectord-1.2.3.cvs.20050128-1.rh.el.um.1.i386.rpm
  - heartbeat-pils-1.2.3.cvs.20050128-1.rh.el.um.1.i386.rpm
  - heartbeat-stonith-1.2.3.cvs.20050128-1.rh.el.um.1.i386.rpm
  - ipvsadm-1.21-1.rh.el.1.um.1.i386.rpm
  - libnet-1.1.2.1-1.rh.el.um.1.i386.rpm
  - perl-Authen-SASL-2.08-1.rh.el.um.1.noarch.rpm
  - perl-Convert-ASN1-0.18-1.rh.el.um.1.noarch.rpm
  - perl-IO-Socket-SSL-0.96-1.rh.el.um.1.noarch.rpm
  - perl-Mail-IMAPClient-2.2.9-1.rh.el.um.1.noarch.rpm
  - perl-Net-SSLeay-1.25-1.rh.el.um.1.i386.rpm
  - perl-Parse-RecDescent-1.94-1.rh.el.um.1.noarch.rpm
  - perl-XML-NamespaceSupport-1.08-1.rh.el.um.1.noarch.rpm
  - perl-XML-SAX-0.12-1.rh.el.um.1.noarch.rpm
  - perl-ldap-0.3202-1.rh.el.um.1.noarch.rpm
  - perl-Digest-HMAC
  - perl-Digest-SHA1
  - perl-Parse-RecDescent
  - iproute

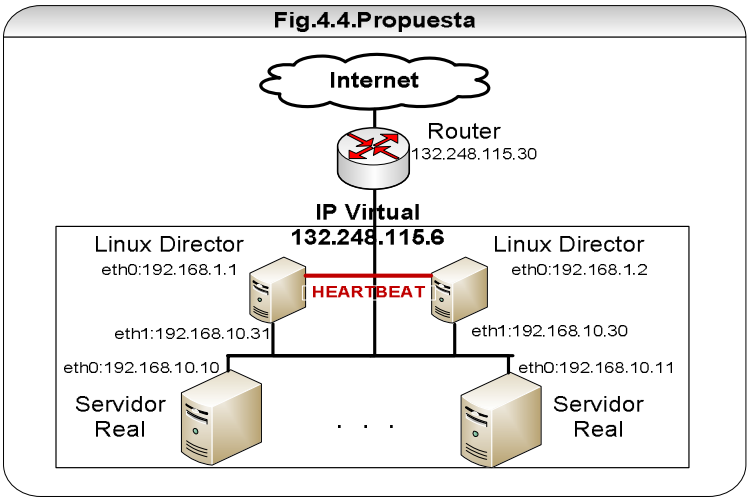
Aura y Azuki

- Sistema Operativo Red Hat Enterprise Linux versión 4, kernel 2.6.9-5.
- Servidor Apache
- Base de Datos Sybase ASE 12.5
- Sybase Replication Server 15.0.1 y Worksheet
- Aplicación Tiendas-UNAM

4.3.6 Aplicaciones

Recordemos que las aplicaciones son el servicio que se brinda, en este caso la página Web y la Base de Datos de Tiendas-UNAM deben estar ambas funcionales en los servidores Aura y Azuki. Con este punto resuelto solo nos debemos ocupar del proyecto de alta disponibilidad.

4.3.7 Red



La IP pública virtual con la que se dará servicio es la 132.248.115.6 cuyo router es la IP 132.248.115.30.

Mopeia se estará monitoreando con Dhori a través de las interfaces de red eth0 con IP's privadas.

Mopeia y Dhori tendrán otra IP privada de otro segmento 192.168.10.0/24, por medio de ésta interfaz de red, eth1, se estará comunicando con los servidores reales cuyas IP's privadas pertenecen a éste mismo segmento

Los servidores reales tendrán configurado como gateway la IP pública virtual 132.248.115.6 e IP's privadas del segmento 192.168.10.0/24.

Mopeia

eth0	192.168.1.1	Mask:255.255.255.0
eth0:0	132.248.115.6	Mask:255.255.255.224
eth1	192.168.10.31	Mask:255.255.255.0
eth1:0	192.168.10.24	Mask:255.255.255.0

Dhori

eth0 192.168.1.2 Mask:255.255.255.0
eth1 192.168.10.30 Mask:255.255.255.0

Aura

eth0 192.168.10.10 Mask:255.255.255.0
gateway 132.248.115.6

Azuki

eth0 192.168.10.11 Mask:255.255.255.0
gateway 132.248.115.6

4.4 CONFIGURACIÓN

En este tema se tratará la configuración que debe tener cada servidor virtual y servidor real.

4.4.1 Configuración de los servidores virtuales

Recordemos que el servidor virtual tiene la tarea de distribuir la carga a los servidores reales. Mopeia será el servidor master y Dhori se encontrará en stand-by.

Iptables como NAT

Mascara para el segmento 192.168.10.0/24 como postrouting y prerouting (el firewall modifica la dirección de origen y destino del paquete sustituyéndola por la propia IP que tiene la máquina en Internet).

/sbin/iptables -t nat -A PREROUUTING -j MASQUERADE -s 192.168.10.0/24
/sbin/iptables -t nat -A POSTROUTING -j MASQUERADE -s 192.168.10.0/24

El iptables para ambas máquinas deben permitir la entrada y salida de paquetes por el puerto 80.

Forward de paquetes

Habilitar IPV4 forwarding cambiando el valor a 1 a la variable net.ipv4.ip_forward en el archivo /etc/sysctl.conf. Para confirmar el cambio utilizamos el comando sysctl –p.

Configuración de Heartbeat

Heartbeat corre sobre los dos directores y se asegura de que la dirección IP virtual esté presente y activa en el servidor virtual master.

Para configurar Heartbeat se debe contar con la configuración de los siguientes archivos:

Archivo /etc/ha.d/ha.cf, APÉNDICE 2 en ambas máquinas este archivo le dice a HeartBeat qué tipos de interfaces va utilizar para comunicarse con el otro nodo del cluster. También define los nodos que van a formar el cluster y los archivos de log donde se registrarán las trazas de la aplicación.

Archivo /etc/ha.d/haresources, APÉNDICE 3, en este archivo se definen los recursos que son gestionados por HeartBeat. Los recursos son script Linux Standars Base (LSB) como los que se usan para arrancar o parar servicios al arrancar el sistema en los diferentes runlevels. Heartbeat buscará estos scripts en estas dos rutas: /etc/rc.d/init.d y /etc/ha.d/resource.d, así que se debe alojar el script al menos en una de las dos rutas.

Archivo /etc/ha.d/authkeys con permisos 600, APÉNDICE 4 en ambas máquinas, en este archivo se define el nivel de seguridad con la que se mandaran información los nodos del cluster Heartbeat.

El monitoreo de los servidores reales es controlado por ldirectord el cual es corrido por el heartbeat. Para configurar ldirectord tenemos que configurar el archivo /etc/ha.d/ldirectord.cf.

Archivo /etc/ha.d/ldirectord.cf , APÉNDICE 5 en ambas máquinas, monitorea los servidores reales. Si un servidor real falla, ldirectord lo excluye del grupo. Cuando vuelva a estar en línea será reinsertado.

Para verificar que ldirector está funcionando

/etc/ha.d/resource.d/ldirectord ldirectord.cf status
ldirectord for /etc/ha.d/ldirectord.cf is running with pid: 3552

Los log de heartbeat y ldirector se encuentran en /var/log/messages

Configuración de ldirectord

Archivo /etc/ha.d/ldirectord.cf APÉNDICE 5 en ambas máquinas

4.4.2 Scripts

1. route APÉNDICE 6
2. route1 APÉNDICE 7

Los scripts route y route1 en conjunto definen el gateway que le dará salida a la nueva máquina que quedará enfrente.

4.4.3 Configuración de Sybase Replication Server en los servidores reales

Configuración de Sybase Replication Server en Aura y Azuki

Para la configuración de Sybase Replication Server y para agregar la Base de Datos al sistema de Replicación se hace uso de la utilidad rs_init.

Cada sección corresponde a una ventana o menú en la utilidad rs_init. Solo se agrega la información necesaria.

- Corriendo rs_init

$SYBASE/$SYBASE_REP/install/rs_init

RS_INIT menu

1. Instalar Replication Server

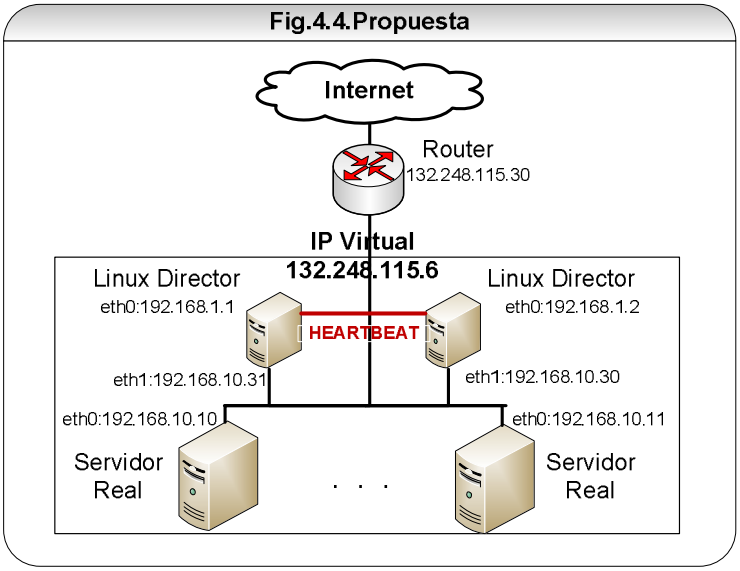
En el APÉNDICE 8 se resumen los datos necesarios para la instalación de Replication Server.

2. Agregar la Base de Datos al sistema de replicación

Se selecciona el Replication Server en el que se encuentra la Base de Datos a replicar.

Se completan los datos con los ya escritos en el APÉNDICE 8, y se seleccionan las Base de Datos.

4.5 FUNCIONAMIENTO DE LA PROPUESTA



La propuesta se basa en la topología de "Alta Disponibilidad y Balanceo de Carga" **Fig.4.4.** del proyecto UltraMonkey.

La propuesta se conforma de dos máquinas que funcionan como servidores virtuales, Mopeia y Dhori, que se encuentran enfrente, ambas máquinas se están monitoreando mutuamente mediante Heartbeat. Mopeia está configurada como master y es la máquina que recibirá las peticiones de los clientes, realizará balanceo de carga con el algoritmo roound robin y las redirigirá a los servidores reales Aura o Azuki, servidores que contienen la aplicación y la Base de Datos, quienes en realidad procesarán la información y devolverán la información ya procesada para ser retornada al cliente; ambos servidores reales monitoreados por ldirectord de tal manera que si un servidor real falla, ldirectord lo excluye del grupo y cuando vuelva a estar en línea será reinsertado. La entrada y salida de las peticiones a través del servidor virtual es posible debido a la configuración realizada de NAT.

La máquina Dhori se encontrará en "espera", ésta máquina estará monitoreando a Mopeia mediante HeartBeat con seguridad en la firma de paquetes del tipo sha1, en el momento en que Mopeia deje de funcionar Dhori tomará su lugar, recordemos que Dhori es una máquina idéntica a Mopeia y que levantará con los mismo servicios y configuración con los que Mopeia cuenta, logrando de ésta manera Alta Disponibilidad en la parte de enfrente.

Ambos servidores reales, Aura y Azuki, tienen repartida la carga de modo que se encontrarán procesando la información requerida por los clientes; en el momento en que uno de estos servidores

falle, el otro servidor real estará dando respuesta a las peticiones. Recordemos que el balanceo de carga también proporciona rapidez de respuesta a peticiones de los clientes.

Debido a que es una aplicación que trabaja con bases de datos, ambos servidores deben tener la misma información; para lograrlo, los servidores reales realizarán replicación de datos entre ellos de forma síncrona mediante Sybase Replication Server. En el momento en que alguna transacción (altas, bajas, cambios) sea completada en cualquiera de los servidores reales, la información será replicada al otro servidor de forma automática.

La Base de datos distribuida es homogenea debido a que los servidores utilizan software idéntico. El sistema cuenta con autonomía local debido a que permite transacciones directas en cualesquier servidor gracias a la replicación.

Es aconsejable la instalación de herramientas extras para el monitoreo de los servicios, en este caso es recomendable la instalación y configuración del MON para tener control sobre el servicio de bases de datos y aplicación Web.

De esta manera el proyecto brinda alta disponibilidad para la aplicación de “Tiendas electrónicas UNAM”.

CONCLUSIÓN

Con el proyecto en marcha la función de los servidores del cluster de alta disponibilidad será estar preparado para entrar inmediatamente en funcionamiento en caso de que falle el otro servidor, ambos servidores se están monitoreando constantemente.

Si se produce un fallo del hardware o de las aplicaciones de alguna de las máquinas del cluster, el software de alta disponibilidad, en este caso heartbeat, es capaz de rearrancar automáticamente los servicios que han fallado en la otra máquina. Y cuando la máquina que ha fallado se recupera, los servicios son nuevamente migrados a la máquina original. Esta capacidad de recuperación automática de servicios nos garantiza la integridad de la información, ya que no hay pérdida de datos, y además evita molestias a los usuarios, que no tienen por qué notar que se ha producido un problema. El cliente no notará ningún tiempo fuera de servicio.

Para finalizar, podemos resumir en tres puntos las importantes ventajas que el proyecto aporta:

1. **Incremento del número de transacciones y velocidad de respuesta ofrecido por los clusters de Balanceo de Carga.** El hecho de contar con una máquina más, de condiciones similares a la máquina que se tiene actualmente, nos ayudará a distribuir la carga, de tal forma que en lugar de ser uno quién responda a todas las peticiones, ahora serán dos servidores los que realicen la tarea. El balanceo de carga se realiza utilizando el algoritmo round robin, es decir, el balanceador distribuye las peticiones de los clientes a los servidores de forma equitativa. Por ejemplo, si tomamos el caso actual en que se está brindando el servicio tendríamos lo siguiente: llegan veinticinco peticiones por segundo, estas peticiones tendrían que ser contestadas por el único servidor que brinda el servicio, este tipo de peticiones son procesos que definitivamente implican una carga en el servidor, ocupan un porcentaje del procesador para llevar a cabo sus tareas y consumen memoria de la máquina; el rendimiento de la máquina es inversamente proporcional a la cantidad de procesos, entre más procesos tenga que realizar la máquina, el rendimiento disminuye y por tanto su velocidad de respuesta. El ejemplo con el sistema implementado sería el siguiente: llegan veinticinco peticiones y el balanceador las reparte entre los servidores reales de acuerdo con el algoritmo round robin, es decir, el servidor AURA en lugar de responder a las veinticinco peticiones, ahora respondería únicamente a trece y el otro servidor AZUKI respondería a doce. De esta forma se impide la sobrecarga de un único servidor distribuyendo los procesos entre ambos, disminuyendo el porcentaje de utilización del procesador y la memoria y por tanto el rendimiento de la máquina, lo cual influye en la velocidad de respuesta debido a que cuenta con mayor disponibilidad de recursos de la computadora para realizar su trabajo.

2. **Incremento de la confiabilidad y robustez ofrecido por los clusters de alta disponibilidad.** Esta característica de alta disponibilidad describe el diseño general del proceso de disponibilidad y la confiabilidad de los equipos y programas. Un proceso robusto y confiable resistirá una variedad de ataques, tanto internos como externos, que podrían fácilmente interrumpir y dañar la disponibilidad en un ambiente más débil. Esto define la disponibilidad del servicio, de acuerdo con lo visto en el capítulo II, un sistema es confiable cuando se eliminan los SPOF, puntos simples de fallo; en este caso, la propuesta ha identificado todos los posibles SPOF que podrían ir contra la disponibilidad del servicio. La propuesta muestra una topología robusta que elimina el SPOF del balanceador de carga (MOPEIA), agregando un servidor de respaldo que se encuentra en stand by (DHORI), en caso de que el balanceador maestro falle; con esto eliminamos el primer SPOF. El segundo SPOF se encuentra en la parte

de los servidores reales, la topología de la propuesta incluye mínimo dos, AURA y AZUKI, éstos servidores contienen la aplicación y son los que brindarán el servicio, éste SPOF ha sido eliminado, cualquier problema que se hubiera podido presentar en ellos está respaldado, de tal forma que si alguno de ellos falla, el otro seguirá brindando servicio; indiscutiblemente el rendimiento de este servidor se verá afectado durante el tiempo que el otro se encuentre fuera de servicio debido a que estará respondiendo a todas las peticiones que se realicen, sin embargo, el servicio jamás se encontrará fuera de funcionamiento. El trabajo en conjunto del software heartbeat, ldirectord, el funcionamiento de LVS y la topología forman un sistema robusto por su complejidad.

3. **Permite aumentar la escalabilidad, disponibilidad y fiabilidad del sistema.** La escalabilidad se refiere a la capacidad de un equipo para hacer frente a volúmenes de trabajo cada vez mayores sin dejar de prestar un nivel de rendimiento aceptable, el sistema proporciona escalabilidad, brinda la posibilidad de agregar uno o más servidores reales dependiendo de las necesidades y problemáticas a las que se enfrente el sistema con el pasar del tiempo. La disponibilidad es la calidad de estar presente, listo para su uso, a mano, accesible, todo esto se logra con la robustez del sistema y la eliminación de SPOF; la fiabilidad es la probabilidad de un funcionamiento correcto a pesar de que hasta el más fiable de los equipos puede fallar, esta situación depende muchísimo tanto del hardware como del software, el mantenimiento y monitoreo constante del equipo de cómputo y el correcto funcionamiento del software instalado.

Inversión:

Como hemos visto en el desarrollo de la presente propuesta, el departamento se encuentra ante una solicitud y una problemática. La solicitud consiste en alta disponibilidad y la problemática es respecto a la saturación del servidor actual que brinda alojamiento a veinticinco aplicaciones Web.

En caso de que no existiera ninguna propuesta como la desarrollada en esta tesis, la opción que el departamento de administración de servidores tomaría sería migrar algunos sitios Web a otra máquina y de esta forma liberar de alguna cantidad de procesos a la máquina actual. Entonces, está haciendo uso de una máquina más, AZUKI.

Mi propuesta incluye ambas máquinas, AURA y AZUKI. Hasta aquí no estoy haciendo ninguna solicitud extra. No perdamos de vista que únicamente se le está dando solución a una problemática.

Esta propuesta solicita dos máquinas más, MOPEIA y DHORI. Una sería el balanceador y la otra se encontraría en stand by. Con la topología propuesta:

- Atendería la petición de Tiendas electrónicas UNAM de brindar un servicio que cuente con alta disponibilidad
- Resolvería la sobrecarga del servidor AURA
- Agregaría alta disponibilidad de las 24 aplicaciones Web restantes que aloja el servidor AURA.

Las ventajas que el proyecto brinda son suficientes para considerar la compra de dos máquinas más cuyos requerimientos son los de un equipo de cómputo tradicional, lo importante es que cuente con mínimo 1 Gb de memoria RAM.

Portal de Tiendas Electrónicas

universidad nacional autónoma de méxico

APÉNDICE 1

DEPENDENCIAS

- Dirección General de Servicios de Cómputo Académico
- Instituto de Investigaciones Jurídicas
- Instituto de Investigaciones Económicas
- Instituto de Investigaciones Sociales
- Dirección General de Divulgación de la Ciencia
- Facultad de Ingeniería
- Dirección General de Estudios de Posgrado
- Facultad de Medicina
- Dirección General de Literatura
- Centro Universitario de Investigaciones Bibliotecológicas
- Instituto de Investigaciones Filosóficas
- Programa Universitario de Estudios de Género
- Instituto de Investigaciones en Matemáticas Aplicadas y en Sistemas
- Facultad de Psicología
- Instituto de Biología
- Facultad de Estudios Superiores Cuautitlán
- Facultad de Contaduria y Administración
- Centro de Investigaciones Interdisciplinarias en Ciencias y Humanidades
- Instituto de Investigaciones Filológicas
- Instituto de Investigaciones sobre la Universidad y la Educación
- Facultad de Economía
- Dirección General de Bibliotecas
- Centro de Investigaciones sobre América Latina y el Caribe
- Oficina del Abogado General
- Facultad de Ciencias Políticas y Sociales
- Centro Regional de Investigaciones Multidisciplinarias

PUNTOS DE VENTA UNAM

- Ventas Directas

APÉNDICE 2
archivo ha.cf

```
#
#     There are lots of options in this file.  All you have to have is
a set
#     of nodes listed {"node ...} one of {serial, bcast, mcast, or
ucast},
#     and a value for "auto_failback".
#
#     ATTENTION: As the configuration file is read line by line,
#               THE ORDER OF DIRECTIVE MATTERS!
#
#     In particular, make sure that the udpport, serial baud rate
#     etc. are set before the heartbeat media are defined!
#     debug and log file directives go into effect when they
#     are encountered.
#
#     All will be fine if you keep them ordered as in this example.
#
#
#       Note on logging:
#       If any of debugfile, logfile and logfacility are defined then
they
#       will be used. If debugfile and/or logfile are not defined and
#       logfacility is defined then the respective logging and debug
#       messages will be loged to syslog. If logfacility is not
defined
#       then debugfile and logfile will be used to log messges. If
#       logfacility is not defined and debugfile and/or logfile are
not
#       defined then defaults will be used for debugfile and logfile
as
#       required and messages will be sent there.
#
#     File to write debug messages to
debugfile /var/log/ha-debug
#
#
#     File to write other messages to
#
logfile     /var/log/ha-log
#
#
#     Facility to use for syslog()/logger
#
logfacility local0
#
#
#     A note on specifying "how long" times below...
#
#     The default time unit is seconds
#           10 means ten seconds
#
#     You can also specify them in milliseconds
#           1500ms means 1.5 seconds
#
#
#     keepalive: how long between heartbeats?
#
#keepalive 1
```

```
#
#     deadtime: how long-to-declare-host-dead?
#
#           If you set this too low you will get the problematic
#           split-brain (or cluster partition) problem.
#           See the FAQ for how to use warntime to tune deadtime.
#
deadtime 5
#
#     warntime: how long before issuing "late heartbeat" warning?
#     See the FAQ for how to use warntime to tune deadtime.
#
#warntime 10
#
#
#     Very first dead time (initdead)
#
#     On some machines/OSes, etc. the network takes a while to come up
#     and start working right after you've been rebooted.  As a result
#     we have a separate dead time for when things first come up.
#     It should be at least twice the normal dead time.
#
#initdead 120
#
#
#     What UDP port to use for bcast/ucast communication?
#
udpport     694
#
#     Baud rate for serial ports...
#
#baud 19200
#
#     serial      serialportname ...
#serial     /dev/ttyS0  # Linux
#serial     /dev/cuaa0  # FreeBSD
#serial     /dev/cua/a  # Solaris
#
#
#     What interfaces to broadcast heartbeats over?
#
bcast eth0        # Linux
#bcast      eth1 eth2   # Linux
#bcast      le0         # Solaris
#bcast      le1 le2           # Solaris
#
#     Set up a multicast heartbeat medium
#     mcast [dev] [mcast group] [port] [ttl] [loop]
#
#     [dev]       device to send/rcv heartbeats on
#     [mcast group]     multicast group to join (class D multicast
address
#                 224.0.0.0 - 239.255.255.255)
#     [port]            udp port to sendto/rcvfrom (set this value to
the
#                 same value as "udpport" above)
#     [ttl]       the ttl value for outbound heartbeats.  this effects
#                 how far the multicast packet will propagate.  (0-255)
#                 Must be greater than zero.
#     [loop]            toggles loopback for outbound multicast
heartbeats.
#                 if enabled, an outbound packet will be looped back
and
#                 received by the interface it was sent on. (0 or 1)
#                 Set this value to zero.
```

```
#
#
#mcast eth1 192.168.10.30 694 1 0
#
#     Set up a unicast / udp heartbeat medium
#     ucast [dev] [peer-ip-addr]
#
#     [dev]       device to send/rcv heartbeats on
#     [peer-ip-addr]    IP address of peer to send packets to
#
#ucast eth1 192.168.10.31
#ucast eth1 192.168.10.31
#
#
#     About boolean values...
#
#     Any of the following case-insensitive values will work for true:
#           true, on, yes, y, 1
#     Any of the following case-insensitive values will work for
false:
#           false, off, no, n, 0
#
#
#
#     auto_failback:  determines whether a resource will
#     automatically fail back to its "primary" node, or remain
#     on whatever node is serving it until that node fails, or
#     an administrator intervenes.
#
#     The possible values for auto_failback are:
#           on    - enable automatic failbacks
#           off   - disable automatic failbacks
#           legacy      - enable automatic failbacks in systems
#                 where all nodes do not yet support
#                 the auto_failback option.
#
#     auto_failback "on" and "off" are backwards compatible with the
old
#           "nice_failback on" setting.
#
#     See the FAQ for information on how to convert
#           from "legacy" to "on" without a flash cut.
#           (i.e., using a "rolling upgrade" process)
#
#     The default value for auto_failback is "legacy", which
#     will issue a warning at startup.  So, make sure you put
#     an auto_failback directive in your ha.cf file.
#     (note: auto_failback can be any boolean or "legacy")
#
auto_failback off
#
#
#       Basic STONITH support
#       Using this directive assumes that there is one stonith
#       device in the cluster.  Parameters to this device are
#       read from a configuration file. The format of this line is:
#
#         stonith <stonith_type> <configfile>
#
#       NOTE: it is up to you to maintain this file on each node in
the
#       cluster!
#
#stonith baytech /etc/ha.d/conf/stonith.baytech
```

```
#
#       STONITH support
#       You can configure multiple stonith devices using this
directive.
#       The format of the line is:
#         stonith_host <hostfrom> <stonith_type> <params...>
#         <hostfrom> is the machine the stonith device is attached
#              to or * to mean it is accessible from any host.
#         <stonith_type> is the type of stonith device (a list of
#              supported drives is in /usr/lib/stonith.)
#         <params...> are driver specific parameters.  To see the
#              format for a particular device, run:
#           stonith -l -t <stonith_type>
#
#
#     Note that if you put your stonith device access information in
#     here, and you make this file publically readable, you're asking
#     for a denial of service attack ;-)
#
#     To get a list of supported stonith devices, run
#           stonith -L
#     For detailed information on which stonith devices are supported
#     and their detailed configuration options, run this command:
#           stonith -h
#
#stonith_host *     baytech 10.0.0.3 mylogin mysecretpassword
#stonith_host ken3  rps10 /dev/ttyS1 kathy 0
#stonith_host kathy rps10 /dev/ttyS1 ken3 0
#
#     Watchdog is the watchdog timer.  If our own heart doesn't beat
for
#     a minute, then our machine will reboot.
#     NOTE: If you are using the software watchdog, you very likely
#     wish to load the module with the parameter "nowayout=0" or
#     compile it without CONFIG_WATCHDOG_NOWAYOUT set. Otherwise even
#     an orderly shutdown of heartbeat will trigger a reboot, which is
#     very likely NOT what you want.
#
#watchdog /dev/watchdog
#
#     Tell what machines are in the cluster
#     node  nodename ...      -- must match uname -n
#node ken3
#node kathy
node  mopeia
node  dhori
#
#     Less common options...
#
#     Treats 10.10.10.254 as a psuedo-cluster-member
#     Used together with ipfail below...
#
#ping 10.10.10.254
#
#     Treats 10.10.10.254 and 10.10.10.253 as a psuedo-cluster-member
#       called group1. If either 10.10.10.254 or 10.10.10.253 are up
#       then group1 is up
#     Used together with ipfail below...
#
#ping_group group1 10.10.10.254 10.10.10.253
#
#     Processes started and stopped with heartbeat.  Restarted unless
#          they exit with rc=100
```

```
#
#respawn userid /path/name/to/run
#respawn hacluster /usr/lib/heartbeat/ipfail

############################
#
#     Unusual options.
#
############################
#
#     hopfudge maximum hop count minus number of nodes in config
#hopfudge 1
#
#     deadping - dead time for ping nodes
#deadping 30
#
#     hbgenmethod - Heartbeat generation number creation method
#           Normally these are stored on disk and incremented as
needed.
#hbgenmethod time
#
#     realtime - enable/disable realtime execution (high priority,
etc.)
#           defaults to on
#realtime off
#
#     debug - set debug level
#           defaults to zero
#debug 1
#
#     API Authentication - replaces the fifo-permissions-based system
of the past
#
#
#     You can put a uid list and/or a gid list.
#     If you put both, then a process is authorized if it qualifies
under either
#     the uid list, or under the gid list.
#
#     The groupname "default" has special meaning.  If it is
specified, then
#     this will be used for authorizing groupless clients, and any
client groups
#     not otherwise specified.
#
#apiauth    ipfail uid=hacluster
#apiauth ccm uid=hacluster
#apiauth ping gid=haclient uid=alanr,root
#apiauth default gid=haclient

# message format in the wire, it can be classic or netstring, default
is classic
#msgfmt  netstring
```

APÉNDICE 3
Archivo haresources

Para Mopeia

```
mopeia 132.248.115.6/27/eth0 ldirectord::ldirectord.cf route
```

Para Dhori

```
dhori 132.248.115.6/24/eth0 ldirectord::ldirectord.cf route
```

APÉNDICE 4
Archivo authkeys

```
#
#     Authentication file.  Must be mode 600
#
#
#     Must have exactly one auth directive at the front.
#     auth  send authentication using this method-id
#
#     Then, list the method and key that go with that method-id
#
#     Available methods: crc sha1, md5.  Crc doesn't need/want a key.
#
#     You normally only have one authentication method-id listed in
this file
#
#     Put more than one to make a smooth transition when changing auth
#     methods and/or keys.
#
#
#     sha1 is believed to be the "best", md5 next best.
#
#     crc adds no security, except from packet corruption.
#           Use only on physically secure networks.
#
auth 2
#1 crc
#2 sha1 HI!
#3 md5 Hello!
2 sha1 hola!
```

APÉNDICE 5
Archivo ldirectord.cf

```
# Global Directives
checktimeout=2
checkinterval=5
#fallback=127.0.0.1:80
autoreload=yes
logfile="/var/log/ldirectord.log"
quiescent=yes

# Virtual Server for HTTP
virtual=132.248.115.6:80
       real=192.168.10.11:80 masq 50
       real=192.168.10.10:80 masq 50
#       fallback=10.1.1.1:http masq 1
#       service=http
#       service=sybase
#       request="index.html"
#       receive="Test Page"
        scheduler=wrr
        persistent=600
#       protocol=
        checktype=negotiate
```

APÉNDICE 6
Script route

```
#!/bin/sh

if [ $# -eq 0 ]; then
 echo "Sintaxis incorrecta"
 echo $0 " [start|stop]"
 exit 1
 else

 case "$1" in

     'start')
/etc/ha.d/resource.d/route1 start &
      ;;

     'stop')
/etc/ha.d/resource.d/route1  stop &
      ;;

      *)

        $0
        ;;

esac

fi
```

APÉNDICE 7
Script route1

```
#!/bin/sh

if [ $# -eq 0 ]; then
 echo "Sintaxis incorrecta"
 echo $0 " [start|stop]"
 exit 1
 else

 case "$1" in

     'start')
      route add default gw 132.248.115.30
      ;;

     'stop')
      route del default gw 132.248.115.30

      ;;

      *)

        $0
        ;;

esac

fi
```

APÉNDICE 8
Instalación de Replication Server

Replication Server information

Replication Server name:*
Is this Replication Server the ID Server?*
Replication Server error log:
Replication Server configuration file:
Replication Server password encryption:
Replication Server language:

Replication Server security information

use Secure Sockets Layer (SSL) security?
If Yes, SSL identity file:
If Yes, SSL private key password (default is `password`):

Replication Server interfaces information
Host name/address:*
Port:*
Name alias:*

ID Server information
ID Server name:*
ID Server user:
ID Server password:
Starting Replication Server ID:
Starting database ID:

Replication Server System Database choice
Will RSSD be embedded? (default is `no`)

Adaptive Server Anywhere Embedded Replication Server System Database information
Complete if you selected Yes for "Will RSSD be embedded?"
ERSSD name:*
ERSSD database file directory:*
ERSSD transaction log directory:*
ERSSD backup directory:*
ERSSD error log directory:*

Adaptive Server Enterprise Replication Server System Database information
Complete if you selected No for "Will RSSD be embedded?"
RSSD Adaptive Server name:*
RSSD name:
Will RSSD be replicated?*
Allow HA failover for RSSD connections?
Create RSSD:*
SA user:

SA password:*
Primary user:
Primary password:
Maintenance login:
Maintenance password:

Adaptive Server Enterprise RSSD device information
Complete if you selected No for “Will RSSD be embedded?”
Size of the RSSD database:
RSSD device name:*
Create the RSSD device:*
RSSD device physical name:
RSSD device size:
Size of the RSSD log:
RSSD log device name:*
Create the RSSD log device:*
RSSD log device physical name:
RSSD log device size:

Disk partition information
Disk partition path:*
Logical identifier for disk partition:*
Size of disk partition:
Start value for partition:

Remote site connections information
Replication Server login name:
Replication Server password:

RSSD RepAgent information
Complete if you selected Yes for “Will RSSD be replicated?”
RS user:
RS password:

BIBLIOGRAFÍA

1. Oscar Rodríguez Fernández, Roberto Troncoso Egea, Sagrario Bravo de Pablo; *La Biblia de Internet;* edición 2006, Madrid, Ed. Anaya multimedia (Grupo Anaya, S.A.).

2. Laura Raya González, Raquel Álvarez Cornejo, Victor Rodrigo Raya; *Sistemas Operativos en entornos: monousuario y multiusuario;* 1a. ed., México, Alfaomega Grupo Editor, 2005.

3. Sebastián Sánchez Prieto; *Unix y Linux guía práctica;* edición 2000, México D.F., Alfaomega Grupo Editor.

4. Elias M. Awad; *Comercio electrónico;* Madrid, Ed. Anaya Multimedia (Grupo Anaya, S.A.), 2007.

5. Óscar Rodrigo González López; *Comercio electrónico;* edición 2006, Madrid, Ed. Anaya Multimedia (Grupo Anaya, S.A.).

6. Edwards, Jeri con Deborah De Voe; *Cliente-servidor: guía de supervivencia;* Traduc. Orfali Robert, 3a. ed., Ed. México: Oxford University, 2002.

7. Ramez Elmasri, Shamkant B. Navathe; *Fundamentos de Sistemas de Bases de Datos;* 3a. ed., Madrid , Ed.Addison Wesley.

8. Rajkumar Buyya; *High performance cluster computing;* Melbourne Australia, Ed. Prentice Hall PTR, 1999.

9. Alex Berson, George Anderson; S*ybase and Client/Server Computing;* New York, Ed. McGraw-Hill, 1995.

10. Coulouris, George F.; *Sistemas distribuidos: conceptos y diseño;* 3a. ed., Madrid, Ed. Addison Wesley, 2001.

11. Chris Oggerino, (2001). *High availability network fundamentals.* Boston, Massachusetts. Libro electrónico consultado en marzo 2008, de libros digitales Safari.

REFERENCIAS

12. Linda G. Lara Vivas, (2006). *Comercio electrónico, la tecnología al servicio del negocio.* ENTER@TE, suplemento de divulgación sobre Cómputo, Internet y Telecomunicaciones [online]. Disponible en Internet vía Web URL: http://www.enterate.unam.mx/Articulos/2006/junio/comercio.htm

13. Sitio oficial designado para actuar como fuente central de información de Linux. Registrado por Michael McLagan [online]. Disponible en Internet vía Web URL: http://www.linux.org. Página Web consultada en junio 2007.

14. Suse Linux Enterprise de Novell, sitio oficial de Suse [online]. Disponible en Internet vía Web URL: http://www.suse.com. Página Web consultada en marzo 2008.

15. Sitio oficial de Debian distribución de Linux, su líder actual es Steve McInntyre [online]. Disponible en Internet vía Web URL: http://www.debian.org. Página Web consultada en marzo 2008.

16. Sitio oficial que brinda información de Red Hat distribución de Linux, compañía y subsidiarios [online]. Disponible en Internet vía Web URL: http://www.redhat.com. Página Web consultada en marzo 2008.

17. Linux High Availability, sitio oficial Alta Disponibilidad [online]. Disponible en Internet vía Web URL: http:// http://www.linux-ha.org. Página Web consultada en junio 2007.

18. Linux Virtual Server, sitio oficial Servidor Virtual Linux [online]. Disponible en Internet vía Web URL: http://www.linuxvirtualserver.org. Página Web consultada en junio 2007.

19. Sitio oficial del proyecto UltraMonkey [online]. Disponible en Internet vía Web URL: http://www.ultramonkey.org. Página Web consultada en junio 2007.

20. Sybase enterprise, sitio oficial ASE_12.5.2 [online]. Disponible en Internet vía Web URL:http://infocenter.sybase.com

21. Sybase enterprise, sitio oficial Replication Server 15.0.1 [online]. Disponible en Internet via Web URL:http://infocenter.sybase.com/help/topic/com.sybase.help.rs_15.0.1/title.htm

22. Jose Angel de Bustos Pérez. *Utilización y administración avanzada de sistemas GNU/Linux y aplicaciones, clustering y alta disponibilidad en GNU/Linux.* [online]. Disponible en Internet como PDF, URL:http://www.augcyl.org/doc/2007-curso-salamanca/SoftwareCientifico.pdf.

23. Juan Pedro Paredes, Alta disponibilidad para Linux, [online] Disponible en Internet como PDF, URL:http://www.ibiblio.org/pub/Linux/docs/LuCaS/Presentaciones/200103hisPalinux/paredes/pdf/LinuxHA.pdf.