Distributed Architectures As Solution to Demanding Systems of Critical Use Services Considering Security, Performance and Availability Factors.

Arquitecturas Distribuidas Como Solución a Sistemas Demandantes de Servicios de Uso Crítico Considerando Factores de Seguridad, Rendimiento y Disponibilidad.

Mario Bonilla, Luis Vargas, Erick Meneses
Universidad Industrial de Santander, Colombia
julian.1221@gmail.com, luivar22@gmail.com, eramenes@uis.edu.co

Abstract

Given the importance and impact that have the availability and performance in information services for network environments with increased demand, distributed architectures are being developed to satisfy these needs. One of them is the high availability cluster, which enables us to orchestrate a set of heterogeneous resources to provide a service for long periods of time without interruption.
This paper presents the design process, technology selection and implementation of a high availability cluster system based around three layers: load balancing, network service and distributed storage, taking into consideration security in each of them and the system in general, obtaining therefore a reliable and scalable system with high levels of performance and availability.

Resumen

Dada la importancia e impacto que toma la disponibilidad y rendimiento en servicios de información para entornos de red con aumento en la demanda de estos, se plantean arquitecturas distribuidas que permitan satisfacer estas necesidades, una de ellas es el cluster de alta disponibilidad, que permite orquestar un conjunto de recursos heterogéneos para prestar un servicio por largos períodos de tiempo sin interrupciones.

Este trabajo presenta el proceso de diseño, selección de tecnologías e implementación de un sistema cluster de alta disponibilidad basando en tres capas: balanceo de carga, servicio de red y almacenamiento distribuido, teniendo en cuenta la seguridad en cada una de ellas y el sistema en general, obteniendo así un sistema confiable y escalable con altos niveles de rendimiento y disponibilidad.

1. Introducción

Actualmente, debido a la masificación en el uso de servicios informáticos nace la necesidad de funcionar de manera ininterrumpida los 365 días del año[1], buscando mejorar la organización y configuración de los componentes que prestan dichos servicios[2]. Las soluciones tradicionales a la necesidad de servicios disponibles han hecho que las organizaciones opten por el uso de sistemas centralizados con altos costos y con inconvenientes de escalabilidad al no soportar demandas crecientes del servicio. Por estas razones, se ha presentado un auge en el desarrollo de conceptos y tecnologías basadas en sistemas distribuidos por medio de las cuales se logran obtener niveles apropiados de disponibilidad, rendimiento, escalabilidad, seguridad y de relación costo beneficio, que conforman soluciones para las organizaciones actuales dada la masificación de la comunicación y el creciente valor de la información. Algunas de ellas son sencillas como en el caso de clusters de balanceo con DNS[15][16] donde no existe un manejo óptimo de la carga, o haciendo uso de balanceadores más específicos para algunos servicios como el módulo para balanceo de carga de apache[17], lo que se convierte en una desventaja a la hora de querer prestar otros servicios como el de correo electrónico.

Para muchas empresas, las interrupciones no planeadas representan tiempo sin el apoyo de sus sistemas informáticos y puede llegar a ser catastrófico, o al menos muy costoso, como lo revelan dos estudios

independientes realizados por Giga Group[1] y Eagle Rock Alliance,Ltd[2], encontrando que una empresa pierde en promedio hasta 6.45 millones dólares por hora de interrupción no planeada de sus servicios informáticos.
Dado el grado de impacto, surgen tecnologías como el cluster de alta disponibilidad (cluster High Availability) que permite obtener sistemas distribuidos tolerantes a fallos, haciendo uso de la redundancia en los componentes de dicha arquitectura, siendo capaz de soportar fallos en el menor tiempo posible y de forma transparente para el usuario. Linux HA Project y Linux Vitual Server (LVS) son los encargados de aunar los esfuerzos de la comunidad de software libre para hacer de GNU/Linux una plataforma en la cual se puedan desarrollar con éxito este tipo de arquitecturas distribuidas.

En las siguiente sección se muestra de manera general la arquitectura de un cluster de alta disponibilidad, sus componentes y la interacción entre ellos dentro del sistema, más adelante se presenta la arquitectura propuesta teniendo en cuenta características de disponibilidad, rendimiento y seguridad, las herramientas tecnológicas que intervienen en el proceso y como se acoplan para alcanzar el objetivo deseado; una vez expuesta la arquitectura se procede a hacer un análisis de disponibilidad y escalabilidad basándose en pruebas y cálculos formales que permitan clarificar la mejora del servicio al utilizar sistemas distribuidos, finalmente se muestran los resultados, conclusiones del trabajo y bibliografía.

2. Arquitectura Cluster de Alta Disponibilidad

Actualmente existen varias implementaciones de clusters de alta disponibilidad y todas ellas buscan ofrecer excelentes niveles de disponibilidad y eficiencia del servicio resguardando siempre la seguridad de la información y haciendo énfasis en procesos de balanceo de carga, almacenamiento y manteniendo la fiabilidad del sistema a través de la redundancia como técnica principal [18]. La mayoría de soluciones sobre cluster de alta disponibilidad son de tipo comercial y muy pocas de tipo académico.

Soluciones comerciales

- IBM High Availability Cluster (HACMP) para AIX y GNU/Linux.
- Red Hat Cluster Suite.
- Microsoft Cluster Server (MSCS) W.server 2008.
- Sun Cluster para Solaris y OpenSolaris.
- Oracle Clusterware para Oracle Real Application Clusters (RAC).
- HP Serviceguard para HP-UX y GNU/Linux.

Soluciones investigativas/académicas

- LVS Linux Virtual Server, soluciones de la comunidad de software libre.
- Linux-HA, paquetes de uso libre para el sistema operativo GNU/Linux.

El proyecto *Linux Virtual Server*[3] proporciona un servidor escalable y altamente disponible construido sobre un sistema balanceador de carga, un cluster de servidores reales y un sistema de archivos distribuido, todo este conjunto corriendo sobre sistemas operativos GNU/Linux.
La arquitectura del cluster es transparente a los usuarios finales, quienes sólo ven un único servidor virtual en una única dirección IP, esto es posible gracias a los métodos de reenvío de conexión que permite que esta serie de recursos puedan atender solicitudes de servicio en conjunto. Los componentes de ésta arquitectura se muestran en la figura 1 donde están agrupados por capas y estas son: capa de monitorización de servicios y balanceo de carga, capa de servidores reales y capa de almacenamiento compartido. A continuación se presenta una descripción de las capas y cada unos de sus componentes.

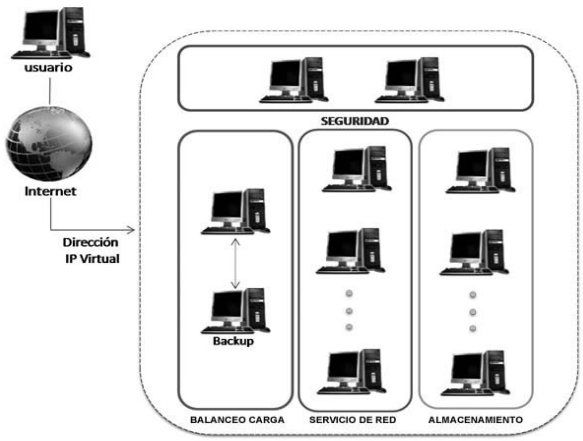


Figura 1: Arquitectura General Cluster Alta Disponibilidad

1 Firma internacional de analistas en tecnología de información. Resultados del estudio se encuentran disponibles en: http://www.dba-oracle.com/art_dbazine_high_avail.htm.

2 Eagle Rock Alliance, Ltd. es una empresa consultora especializada en Fiabilidad de consultoría y tecnología. Resultados del estudio: http://www.beechtek.net/page/1012471

2.1 Capa de Monitoreo y Balanceo de Carga

Las peticiones de servicio al servidor virtual son recibidas por un conjunto de máquinas denominado sistema balanceador (a través de una dirección IP virtual VIP), quien estará en continua monitorización de los servicios prestados y redirigirá las peticiones a un conjunto de servidores reales que darán una respuesta a los usuarios del sistema. Esta capa está compuesta por dos nodos y se permiten dos posibles configuraciones para su funcionamiento. En la primera denominada Activo/Activo los dos nodos reciben peticiones de servicio, en la segunda llamada Activo/Pasivo existe un nodo primario o principal que recibe peticiones de servicio y otro secundario o backup el cual permanece en estado de espera hasta que ocurra un fallo en el primario o principal para retomar los servicios prestados por este último.

Monitorización de servicios

Una de las herramientas software provistas por el proyecto Linux HA3 es heartbeat cuya labor es mantener servicios altamente disponibles. Este basa su funcionamiento en el envío y recepción de "latidos"; señales enviadas por los demonios *heartbeat* que corren en las máquinas del sistema balanceador. Además, heartbeat es el encargado de lanzar la recepción del servicio que ha de prestarse en alta disponibilidad y activar la VIP en el nodo secundario, cuando detecta la ausencia de latidos por parte del nodo balanceo primario.

Balanceo de carga

El proceso de balanceo de carga hace uso de la lista de servidores reales disponibles que se obtienen mediante una herramienta de administración llamada Ldirectord la cual en tiempo real asigna unos recursos a una solicitud de servicio dependiendo del algoritmo de planificación.
Ldirectord (Linux Director) es un demonio lanzado por heartbeat para la monitorización y control de los servidores reales, así como para llevar a cabo la administración de las tablas de forwarding del kernel. Ldirectord verifica el estado de los servidores reales en intervalos de tiempo predeterminados permitiendo actualizar dinámicamente las tablas que definen el comportamiento del balanceador de carga IPVS, así si uno de los servidores reales deja de atender peticiones, ldirectord creará una llamada a ipvsadm que hará que el balanceador deje de reenviarle paquetes al servidor real.

Algoritmos de planificación

El sistema balanceador es el encargado de distribuir la carga entre los diferentes servidores reales. Para hacerlo se basa en conjunto secuencial de normas, llamadas algoritmos de planificación (scheduling), que permiten decidir en cada momento a que servidor debe reenviar los paquetes pertenecientes a una misma conexión. Dentro de los algoritmos que se pueden usar con el proyecto LVS[3] se encuentran :

- Round-Robin (rr)
- Weighted Round-Robin (wrr)
- Least-Connection (lc).
- Weighted Least-Connection (wlc).

2.2 Capa de Servicio

En esta capa se encuentran los servidores reales los cuales son los encargados de procesar las peticiones de servicio redirigidas desde el sistema balanceador de carga. Entre los servicios que se pueden prestar están: servicio web, correo electrónico, bases de datos, VoIP y aplicaciones más específicas como un CRM o una ERP entre otros.

En este proyecto se seleccionaron dos servicios (servicio web y servicio de correo electrónico) a fin de brindar diferentes escenarios para analizar el comportamiento de la arquitectura propuesta.

2.3 Capa de Almacenamiento Compartido

Sistemas de almacenamiento compartido

El incremento de la demanda de cómputo en ambientes científicos, comerciales ha llevado a una alta demanda no solo de poder computacional, sino de capacidad de almacenamiento para ubicar la información ya procesada o datos a procesar. Se observa a continuación las características más relevantes que tienen los sistemas de archivos, tomadas en cuenta en la arquitectura propuesta para cumplir con las necesidades de compartir y unir todos los discos como uno solo sin dejar de lado los dos objetivos principales de esta propuesta que son: el rendimiento (paralelismo) y la tolerancia a fallos (replicación).

---

3 Proyecto Linux-HA tiene por objetivo brindar una solución en alta disponibilidad para servicios de red sobre sistemas GNU/Linux http://www.linux-ha.com/

Sistemas de archivos distribuidos

En un cluster de alta disponibilidad los servidores reales necesitan tener acceso y manipular la misma información, una solución a esta necesidad es mediante los sistemas de archivos distribuidos, en donde varios usuarios comparten archivos y almacenan recursos.

Las ventajas de los sistemas de archivos distribuidos respecto a los sistemas centralizados son [4]:

- Compartir recursos: los usuarios tienen acceso a sus recursos compartidos a través de la red.
- Reducción de costos: Las redes de equipos de cómputo ofrecen una mejor relación precio/rendimiento que los mainframes.
- Tolerancia a fallos: Los sistemas distribuidos permiten replicar recursos y procesos, a fin de proporcionar fiabilidad y disponibilidad al sistema, el fallo de un nodo no implica el fallo global.

Algunos sistemas de archivos distribuidos son: NFS[5] (Network File System) y GlusterFS[6].

Sistemas de archivos paralelos

En los sistemas de archivos distribuidos, los archivos se almacena en un servidor, y el ancho de banda de acceso a un archivo se encuentra limitado por el acceso a un único servidor. Este problema, es originado por el desequilibrio existente entre el tiempo de cómputo y el tiempo de E/S (Entrada/Salida).[7]

Lo anterior situación se soluciona con los sistemas de archivos paralelos, puesto que estos utilizan paralelismo en el sistema de E/S con la distribución de los datos de un archivo entre diferentes dispositivos y/o servidores. Esto evita los cuellos de botella en los procesos de E/S ya que se accede de forma paralela a un archivo, aporta en el rendimiento al hacer un mejor uso del ancho de banda del sistema total. Algunos sistemas de archivos paralelos son: Lustre[8], Parallel Virtual File System PVFS2[9].

Tolerancia a fallos

Tolerancia a fallos es la capacidad de un sistema de mantenerse en funcionamiento ante un fallo. Disminuir la probabilidad de fallo está relacionada con la implementación de técnicas como la replicación de componentes en el sistema. Si algún componente falla otros pueden responder correctamente, haciendo de este proceso transparente para el usuario (failover)[10].

Los mecanismos para tolerancia a fallos en almacenamiento pueden tener un soporte hardware o software, o bien una combinación de ambos. Un modulo del kernel de GNU/Linux es DRBD (Distributed Replicated Block Device) el cual permite hacer replicación remota en tiempo real en dispositivos de almacenamiento por medio de TCP/IP (RAID1)[11].

2.4 Capa Transversal de Seguridad

La seguridad es una característica deseable en todo sistema y su aplicación un cluster, donde su fuerte es el rendimiento obtenido a través de la distribución de recursos y trabajos, es también su problema, inherente a su naturaleza distribuida que va desde la simple transferencia de archivos hasta operaciones complejas de colaboración y resolución de problemas.

Es importante resguardar la seguridad garantizando la integridad, confidencialidad y disponibilidad del sistema [21], en este sentido es posible abordar el problema de forma perimetral descendiendo desde el nivel de red, nivel de host hasta el nivel de servicio, valiéndose de las herramientas proporcionadas en proyectos reconocidos, como: Netfilter[4], Squid[5], Snort[6], SELinux[7] y PKI[8] entre otros.

2.5 Proceso de comunicación entre el usuario y el servicio.

La necesidad de atender y responder las solicitudes de servicio de manera transparente para el usuario requiere de métodos de reenvío de conexiones que hacen uso de las propiedades técnicas y geográficas de la topología de red que se use. Los métodos propuestos por LVS [3] se listan a continuación:

- Método de reenvío de conexiones por NAT (VS-NAT).
- Método de reenvío de conexiones por encapsulado IP (VS-Tun).
- Método de reenvío de conexiones por enrutamiento directo (VS-DR).

4 Framework contenido en el sistema operativo linux para el filtrado de paquetes de red http://www.netfilter.org/

5 Servidor Proxy para filtrado de paquetes y aceleración de servicios web haciendo uso de almacenamiento en cache http://www.netfilter.org/

6 Sistema de detección de intrusos http://www.snort.org/

7 Conjunto de módulos de seguridad que provee el sistema operativo Linux para la implementación de políticas de acceso y control.

8 Infraestructura de Llave Publica, mecanismo de autenticación usado en sistemas distribuidos http://www.pki-page.org/

3 Disponibilidad en Sistemas Distribuidos

Un dilema común en las TIC's es cómo sobrevivir a interrupciones planeadas y no planeadas, por medio de los los sistemas distribuidos se logra distribuir la probabilidad de fallos, en la práctica, esto se requiere del cuidado en el diseño, configuración e integración de los componentes que hacen parte del mismo. En un entorno distribuido, los criterios básicos para calcular la disponibilidad son los siguientes [12]:

- Determine la disponibilidad de cada componente, como un número de 0 a 1.
- Multiplicar la disponibilidad de todos los componentes juntos para conseguir la disponibilidad total, que generalmente se expresa como un porcentaje.

En un cluster de alta disponibilidad se pueden identificar 3 componentes que conforman una arquitectura de este tipo, los cuales son: balanceadores de carga, nodos del servicio prestado y el almacenamiento compartido.

Matemáticamente, esto se describe como [12]:
$D_{total} = D_{BalanceoCarga} * D_{NodosServicio} * D_{Alm.Compartido}$ *(2)*
$D_{total} = D_{total} * 100\%$ *(3)*
D = Disponibilidad

En el contexto de disponibilidad existen dos formas de clasificar los componentes computacionales de un cluster, estos son:

- Cluster de nodos Activos: Conjunto de componentes computacionales que se encuentran prestando su servicio activamente todo el tiempo.
- Cluster de nodos Activo/Pasivo: Conjunto de componentes computacionales que tienen un componente igual asociado, el cual es pasivo y monitorea al nodo activo para en caso de falla retoma los servicios del primero.

Fallo del cluster

Considerando un cluster de nodos Activo/Pasivo, si falla un nodo el otro retomara los servicios, sin embargo si fallan los dos nodos el servicio estará no disponible. Se definen los siguientes parámetros para el cálculo de la disponibilidad del sistema:

n = número de nodos del sistema.
a = disponibilidad de un nodo (porcentaje del tiempo que está funcionado correctamente).
f = probabilidad de fallo de un nodo (porcentaje del tiempo que no funciona).
F_d = probabilidad de fallo del sistema.
F = probabilidad de que el sistema este dañado.
A= disponibilidad del sistema.

Siendo 1 el 100% se deduce que la probabilidad de no prestar el servicio es $f = 1- a$, luego la probabilidad de que 2 nodos no estuvieran prestando el servicio al mismo tiempo es la probabilidad de que un nodo no funcione y la probabilidad que el segundo nodo tampoco funcione, matemáticamente esto es $f = 1–a^2$ enmarcado en un cluster de ^{n} nodos seria $f = 1–a^n$. El número de formas (^{c}) en las que pueden fallar los componentes del cluster depende de la manera como fue organizado, más adelante se analiza la variable ^{c} para cada caso especifico en la organización de los componentes del cluster de alta disponibilidad.
Por último la probabilidad de fallo del cluster es [13]:
$Fd = c*(1-a)^n$ *(4)*

Eventos de Failover

En el caso de que un nodo falle el sistema se mantiene en funcionamiento, sin embargo hay un tiempo usado para la retoma de servicios, este es el tiempo que transcurre desde que un nodo activo es reportado como dañado, hasta que su copia (nodo pasivo) se convierte en activo y retoma los servicios. Se define lo siguiente:

F_f = Probabilidad que el sistema no funcione por causa de un proceso de failover.
MTFO = Tiempo medio de failover (tiempo promedio de retoma de servicios en caso de fallo)
MTFB= Tiempo medio entre fallos de un nodo.

La proporción de tiempo que el sistema esté dañado durante un failover de un nodo en particular es *MTFO/MTFB*, teniendo en cuanta que hay ^{n} nodos en el sistema, el fallo de cualquier nodo llevara asociado un evento de failover. La probabilidad que el sistema no funcione durante el proceso de failover es [13]:
$F_f = n*(MTFO/MTFB)$ (5)

Disponibilidad en un cluster

La probabilidad que el sistema no funcione (F) es la probabilidad de fallo del cluster (F_d) más la probabilidad que el sistema no funcione por causa de un proceso de failover (F_f) [13]:
$F = F_d + F_f = c*(1-a)^n + n(MTFO/MTBF)$ *(6)*
La disponibilidad del sistema (D) es:
D = 1 – F (7)

4 Arquitectura Propuesta

La arquitectura propuesta fue diseñada con la unión de componentes que aportan características de escalabilidad, rendimiento, seguridad y tolerancia a fallos mediante la replicación de servicios e información. Las tres capas verticales que se muestran en la figura 2 definen el orden en que una petición de servicio será atendida por el servidor virtual y su integración será descrita en esta sección.

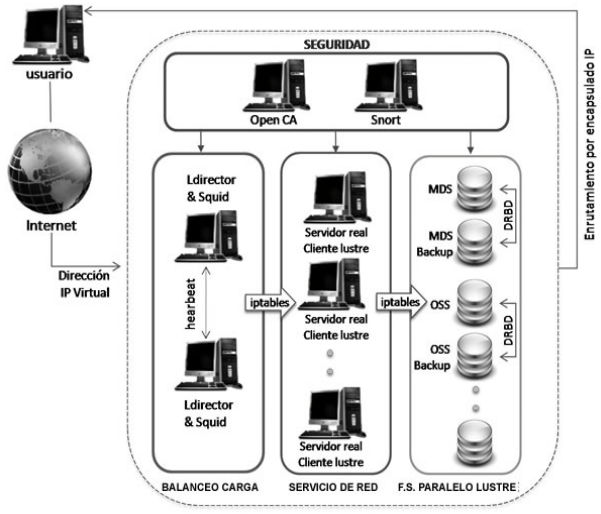


Figura 2: Arquitectura Propuesta

Capa de Balanceo

Esta encargada de recibir las peticiones de servicio y de repartirlas en el conjunto de servidores reales. Al estar compuesta por dos balanceadores su probabilidad de fallo que se distribuye pues mientras que un nodo está procesando las solicitudes el otro se encuentra monitoreando (configuración Activo/Pasivo) para en caso de falla retomar el servicio de manera rápida sin que el usuario note la ausencia del mismo.
El proyecto Linux-HA provee el software encargado de la monitorización: heartbeat, el cual monta automáticamente la interfaz con la IP virtual del servidor para recibir las peticiones de servicio y, cuando el nodo principal falla, el demonio hearbeat será el encargado de levantar una interfaz con la IP virtual en el nodo secundario donde se seguirán prestando los servicios del principal garantizando la continuidad del negocio.

En esta misma capa se lleva a cabo el balanceo de carga, tarea realizada por el demonio ldiretord el cual hace uso de reglas IPVSadm para actualizar la tabla de servidores reales disponibles para prestar servicio. El algoritmo de planificación para el balanceo de carga es el WLC (Servidor con menos conexiones activas ponderado), el cual permite asignar un peso a los servidores reales según sus capacidades o según las características de los servicios prestados. Este método garantiza una distribución de las peticiones de servicio según el estado de las conexiones activas en cada servidor evitando la saturación de los mismos.

Capa de Servicio

Para evaluar la arquitectura propuesta se generaron dos escenarios. En el primero se implemento el servicio de páginas web mediante el protocolo HTTPS (Hypertext Transfer Protocol Secure) y el servidor Apache, y en el segundo se implemento el servicio de correo web a través los protocolos SMTP (Simple Mail Transfer Protocol) para intercambio de mensajes, IMAP (Internet Message Access Protocol) para el acceso a mensajes electrónicos en el servidor y las herramientas SpamAssassin[9] para filtrado de paquetes, el antivirus Clamav[10] y Rouncube[11] como cliente Mail-Web, todo bajo comunicación cifrada soportada por medio del protocolo SSL (Secure Socket Layer).

Capa de Almacenamiento

El sistema de archivos basado en cluster que se usó fue Lustre File System, el cual se eligió basándose en los resultados obtenidos de las pruebas de rendimiento donde se comparó con PVFS2 (Parallel Virual File System 2), además de la revisión bibliográfica realizada donde se encuentra que Lustre tiene mejores niveles de rendimiento y escalabilidad al lado de otros sistemas de archivos [8][14].
La distribución de los componentes del sistema de archivos en la arquitectura diseñada es como se muestra en la figura 2, donde un nodo de almacenamiento cumple con las labores de servidor de metadatos MDS y labores de administración del sistema de archivos con el sistema de gestión MGS, además existen 2 nodos de almacenamiento (OSS) que aportan un volumen físico al sistema de archivos paralelo y al igual que el servidor administrador y de metadatos (MDS y MGS) tienen sus respectivas copias de respaldo por medio de DRBD. Esta integración permite obtener un sistema de archivos de alto rendimiento por medio del uso de paralelismo en la E/S, altamente escalable, y tolerante a fallos

9 http://spamassassin.apache.org
10 www.clamav.net
11 http://roundcube.net

mediante la replicación en tiempo real permitiendo que si un servidor del cluster de almacenamiento falla, otro tome su lugar de forma transparente sin que el usuario lo note.

Capa de Seguridad

La seguridad vista de forma perimetral permite asegurar el sistema formando anillos concéntricos de afuera hacia adentro protegiendo la información en varios niveles: red, aplicación y host.

A nivel de red se implenta el filtrado de paquetes a través de iptables lo que permite disminuir el trafico restringiendo paquetes no deseados y delimitando la comunicación solo entre capas adyacentes, lo que disminuye la posibilidad de una intrusión. También se incluye un sistema NIDS (Network Intrusion Detection System) usando la herramienta Snort que brinda auditabilidad al tener un registro centralizado de monitoreo de paquetes malformados en la red y detectados como posibles ataques.
A nivel de aplicación se aprovecha que los servicios implementados se basan en web y se implementa un servidor proxy replicado sobre los balanceadores, de este modo cada paquete que ingresa al cluster es analizado y almacenado en memoria cache lo que brinda seguridad y rendimiento al tener acceso rápido a páginas que no cambian frecuentemente. También se agrega un elemento externo que es la autoridad de certificación implementada a través del software OpenCA[12] a fin de emitir certificados digitales a cada uno de los usuarios del sistema y de este modo establecer una autenticación multifactor basada en login/password y certificado digital. Por último, se garantiza la integridad de la información haciendo uso de protocolos de transferencia cifrada como ssl y ssh, evitando el sniffing en la red.
A nivel de host se realiza el respectivo hardening del sistema operativo así como del servicio que se esté prestando, también se hace uso de SELinux que mediante una correcta configuración permite establecer políticas de división y acceso a los recursos del sistema.

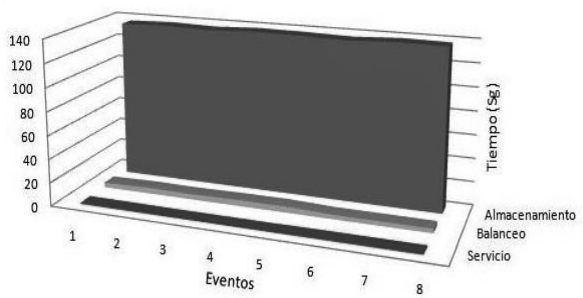
Generalidades

Todo el software usado en la construcción de la arquitectura es software libre y de código abierto disponible en los repositorios Linux o en los portales de cada proyecto. Específicamente, la distribución usada en la implementación fue Debian en su rama estable perteneciente el proyecto GNU/Linux, la cual tiene un índice de disponibilidad de 99,942 (5.08 horas no disponible al año) según un estudio[13] realizado por el Yankee Group Research, Inc. lo que la convierte en una distribución óptima para el desarrollo de sistemas de alta disponibilidad.

A nivel de operación la principal característica es la escalabilidad, que está presente en todos los niveles. A nivel de balanceo hertbeat permite tener un cluster de maquinas recibiendo peticiones y redireccionandolas de acuerdo a el algoritmo de balanceo configurado, en el nivel de servicio se encuentra el conjunto de maquinas que atienden las peticiones propiamente y ofrecen el servicio solicitado, y más abajo se encuentra el nivel de almacenamiento administrado por el sistema de archivos lustre que distribuye la información hasta en 100 nodos dependiendo de la capacidad, que está ligada el método de enrutamiento del sistema balanceador.

5 Calculo de la Disponibilidad

Para el cálculo de la disponibilidad se necesita conocer dos datos característicos de la arquitectura propuesta, el primero de estos es el MTBF de cada máquina y hace referencia a las propiedades de los equipos de cómputo en términos de disponibilidad, tanto en la parte de hardware como en la parte del software instalado. El segundo dato necesario es el tiempo que se tarda en retomar los servicios las copias existentes en las diversas capas de la arquitectura propuesta.



	1	2	3	4	5	6	7	8
Servicio	0,98	0,84	0,58	0,58	0,59	0,84	0,86	0,78
Balanceo	4,1	4,9	4,5	5,1	5,2	3,8	3,8	3,7
Almacenamiento	136,2	137,4	136,3	137,9	137,4	136,4	138,2	137,6

Figura 3: Failover en las diferentes capas del cluster

12 Software que implementa una autoridad de certificación para una infraestructura de llave publica http://www.openca.org/

13 Resultados disponibles en: http://www.iaps.com/2008-server-reliability-survey.html

Las pruebas de failover se realizaron creando eventos de fallos en algunos nodos por capas sacándolos del cluster y midiendo el tiempo que se tardaba en retomar los servicios al nodo que tenía replicado el servicio adjunto. Este proceso se realizó durante 8 veces y se promediaron los resultados.

Primero se realizó la prueba en la capa de balanceo con un tiempo promedio de 4.351 segundos en la retoma del servicio, en la capa de cluster de servidores el promedio fué de 0.762 y en la capa de almacenamiento el tiempo de reconfiguraron del sistema de archivos realizado por el MGS de Lustre FS para incluir la copia proporcionada por DRBD, mostró el tiempo más alto de failover en todas las capas, con un promedio de 137.162 segundos.

5.1 Disponibilidad en la Arquitectura Propuesta

A continuación se realiza el cálculo de disponibilidad en cada una de las capas de servicio de la arquitectura implementada, teniendo en cuenta las ecuaciones expuestas en la sesión 3 y los valores de tiempo antes calculados.

Disponibilidad en la Capa de Balanceo

Con n=2 nodos, a=0,99911[14] , $MTFO_{BalanceCarga}$=4,351 Seg, MTBF=8640 h[15] y c=1 se calculan las ecuaciones (4) (5) (6) (7) obteniendo:

Fd=7.76853e-7, Ft=2.797710e-7, F=1.056624e-6

$D_{BalCarga}$ = 9,999989434e-1

Disponibilidad en la Capa de Servicio

Con n=4 nodos, a= 0,999118, $MTFO_{ClusterServicio}$ = 0,762 Seg, MTBF=8640 h y c=1. Teniendo F_t = *MTFO/MTFB* para sistemas con todos los nodos activos y las ecuaciones (4) (6) (7) se obtiene:

Fd=6.03501e-13, Ft=2.44984e-8, F=2.449906e-8

$D_{ClusterServicio}$ = 9,99999755e-1

Disponibilidad en la Capa de Almacenamiento

Con n=6 nodos, a= 0,999118, $MTFO_{Alm.Compartido}$= 137.162 Seg, MTBF=8640 h y c=1. En el sistema de archivos Lustre replicado, si dos nodos fallan el sistema de archivos no funcionara, la variable c muestra que si hay un cluster de n nodos existe *n*(n+1)/2* combinaciones únicas de un fallo doble de nodos. Con estos parámetros y las ecuaciones (4) (5) (6) (7) se obtiene:

Fd=1.165280e-5, Ft=2.645871e-5, F=3.81115e-5

$D_{AlmCompartido}$ = 9,999618885e-1

Teniendo en cuenta la ecuación (1) enunciada en la sección 3 y reemplazando los valores hallados se tiene:

$D_{total} = D_{BalanceoCarga} * D_{NodosServicio} * D_{Alm.Compartido}$ *(1)*

*Dtotal=9.999989434e-1*9,99999975e-1*9.9996188e-1*

Dtotal =9,99960807e-1

$Disponibilidad_{Total} = 99.99608\%$

Con la arquitectura diseñada e implementada se logró una disponibilidad de cuatro nueves, se puede decir que a lo largo de un período de un año el sistema estará fuera de servicio por 20 minutos 36 segundos. Valores de disponibilidad mayores a 99% son considerados sistemas de alta disponibilidad, y al alcanzar un valor de cuatro nueves la arquitectura propuesta es aceptada para prestar servicios de misión crítica [19]. Es de resaltar la redundancia como característica principal en cada una de las capas mejorando la disponibilidad de cada una de ellas distribuyendo la probabilidad de fallos en las copias activas o pasivas existentes en cada capa de la arquitectura implementada.

5.2 Escalabilidad en la Arquitectura Propuesta

Las pruebas de escalabilidad fueron realizadas sobre un cluster homogéneo de 12 pc's distribuidos de la siguiente manera: 8 pertenecientes al cluster de almacenamiento con un servidor de metadatos y administrador del sistema de archivos Lustre y 3 servidores de almacenamiento, todos con sus respectivas copias; 4 servidores web corriendo apache, que a la vez son clientes del sistema de archivos y 2 servidores balanceadores en configuración activo/pasivo.

14 Valor de la disponibilidad como resultado de cálculo realizado a una máquina Dell Optiplex gx620 con sistema operativo GNU/Linux Debian 4.0

15 Valor suministrado por el Centro de Tecnologías de Información y Comunicación CENTIC-UIS para máquinas Dell Optiplex gx620

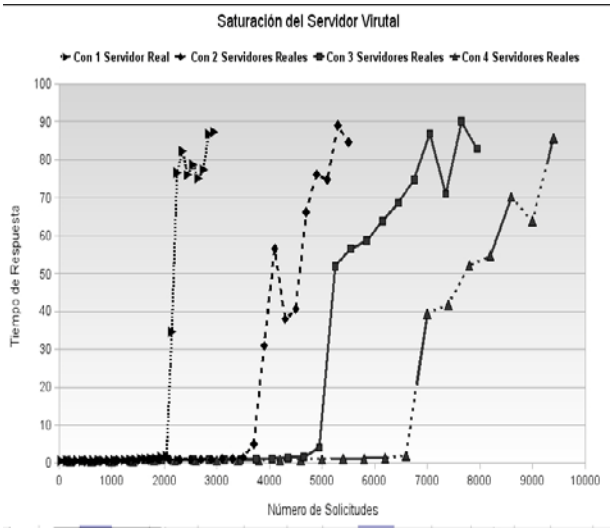


Figura 4: Saturación del Servidor Virtual

El benchmark que se utilizo para las pruebas fue autobench[14], este permite realizar un número de conexiones a una IP solicitando una página con una tasa de solicitud constante hasta alcanzar un número de conexiones establecido. Los resultados generados por el benchmark muestran el comportamiento que se obtuvo en el test ejecutado, algunas de estas medidas son: el tiempo promedio en el cual se atendieron las solicitudes realizadas, el ancho de banda usado en la E/S, etc.

El conjunto de pruebas se realizaron desde la red local con interfaz fast ethernet 10/100, con los resultados se puede medir el punto de saturación del servidor virtual, el cual es aquel donde el tiempo de respuesta empieza a aumentar (>2 segundos) por el trabajo al que está siendo sometido el servidor.

El test se realizó usando escenarios de 1, 2, 3 o 4 servidores reales, en la figura 4 se puede ver la comparación de estos escenarios, en donde se puede observar que el punto de saturación aumenta cuando se aumenta el número de servidores reales garantizando la escalabilidad del servicio cuando la demanda del mismo lo requiera y además asegurando que el servicio http esté disponible por medio de la replicación del servicio que existe en cada servidor, si se daña uno habrá otro(s) que seguirán prestando el servicio.

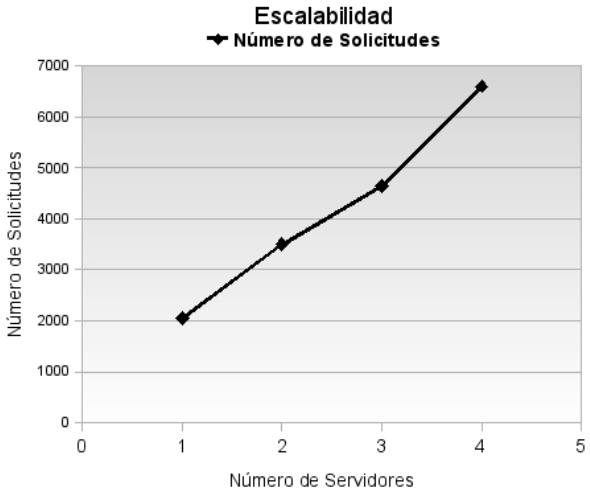


Figura 5: Escalabilidad del Servidor Virtual

El comportamiento lineal que se obtiene entre número de servidores reales y el número de conexiones atendidas muestra la escalabilidad que se obtiene en poder de cómputo al aumentar el número de servidores reales y obtener un incremento del número de conexiones atendidas (punto de saturación).

6 Conclusiones

El diseño de la arquitectura propuesta divide los servicios por capas y replica los componentes en cada una de ellas, disminuyendo la probabilidad de fallos en el sistema.

Se demuestra que demuestra que a través de un correcto diseño e integración de software libre y hardware convencional se puede construir una arquitectura distribuida tipo cluster, logrando alcanzar un índice de disponibilidad aproximado de 99,996% en el servicio prestado.

A partir de las pruebas realizadas se observa que la arquitectura propuesta es altamente escalable, lo que permite satisfacer de manera eficiente el creciente número de solicitudes realizadas por los usuarios con un aumento proporcional en el número de servidores reales que componen el cluster.

El comportamiento del sistema como un todo se ve afectado por los componentes de comunicación (interfaces y medios), que determinan los tiempos de trasmisión de datos y respuestas, de acuerdo al ancho de canal de banda.

7 Referencias

[1] E. Marcus y H. Stern. Blueprints for high availability (2nd Ed.). Capítulo 1: Introduction, Pag 1-5. John Wiley and Sons, New York (2003).

[2] Charles Bookman. 2003. Linux clustering: building and maintaining Linux clusters. Tercera edición. Capitulo 5

[3] Zhang W, 2000. Linux Virtual Servers for Scalable Network Services, Linux Symposium (July. 2000) DOI=http://www.linuxvirtualserver.org/ols/lvs.ps.gz.

[4] Schroeder, M. D. 1993. A state-of-the-art distributed system: computing with BOB. In *Distributed Systems (2nd Ed.)*, S. Mullender, Ed. Acm Press Frontier Series. ACM Press/Addison-Wesley Publishing Co., New York, NY, 1-16.

[5] R. Sandberg, D. Goldberg, S. Kleiman, D Walsh, B. Lyon. 1985 Design and implementation of the SUN network filesystem. In Proceedings of the 1985 USENIX Conference. (June. 1985).

[6] Zresearch. GlusterFS Wiki. 2008. DOI= http://gluster.org/docs/index.php/GlusterFS.

[7] Y.N. Patt. 1994. The I/O subsystem: A candidate for
[8] improvement. 27, 3(Match 1994), 15–16.

[9] Cluster File Systems Inc. Lustre file system. DOI= http://manual.lustre.org

[10] P. Carns Walter, R. B. Ross Rajeev. 2000. PVFS: A Parallel File System for Linux Clusters Parallel Architecture. Research Laboratory Clemson University, Clemson. Volume 2000, Issue 80 (Nov. 2000).

[11] Cristian, F. Understanding fault-tolerant distributed systems. *Commun. ACM* 34, 2 (Feb. 1991), 56-78. DOI= http://doi.acm.org/10.1145/102792.102801

[12] Pla, P. 2006. Drbd in a heartbeat. Linux J. 2006, 149 (Sep. 2006), 3.

[13] Robinson R. y Polozoff A. Planning for Availability in the Enterprise, IBM WebSphere Developer Technical Journal (Dec 2003) DOI=http://www.ibm.com/developerworks/Websphere/techjournal/0312_polozoff/polozoff.html.

[14] Calculating Availability - Cluster Availability, Availability Digest, (May 2007) DOI=http://www.availabilitydigest.com/private/0205/calculating_availability_clusters.pdf

[15] Autobench DOI= http://www.xenoclast.org/autobench/

[16] Bryhni, H. Klovning, E. Kure, O. A comparison of load balancing techniques for scalable Web servers. IEEE (2000).

[17] G. Díaz, J. Chaves, Mendoza, H. Hoeger, L. Núñez. Adaptación de Clusters de Linux para Servicios de Redes. VI jornadas Científico Técnicas de la facultada de ingenierías, Universidad de los Andes (2007).

[18] K. Coar y R. Bowen. Apache Cookbook. Capítulo 10: Proxies, Load Balancing with mod_proxy_balancer. Páginas 211,212. (2007).

[19] E. Marcus y H. Stern Blueprints for high availability (2nd Ed.). Capítulo 9: Data and Web Service, Pag 333-359. John Wiley and Sons, New York (2003).

[20] E. Marcus y H. Stern Blueprints for high availability (2nd Ed.). Capítulo 2: What to Measure, Pag. 9–30. John Wiley and Sons, New York (2003).

[21] M. Pourzandi et Al. Clusters and Security: Distributed Security for Distributed Systems. In 2005 IEEE International Symposium on Cluster Computing and the Grid, 96 -104.