Propuesta de arquitectura de despliegue de clústeres de servidores web de altas prestaciones

Adrian Hernández Yeja
Universidad de las Ciencias Informáticas, La Lisa, La Habana, Cuba, ayeja@uci.cu

RESUMEN

El entorno web se torna cada vez más complejo y competitivo. Un aspecto importante en el posicionamiento de un sitio web lo constituye su rendimiento para satisfacer adecuadamente las demandas de los usuarios. Es por ello que se hace necesario adoptar mecanismos que permitan la optimización del despliegue de sitios web utilizando técnicas y herramientas de probada efectividad. En este trabajo se pretende realizar una propuesta de Software Libre para una arquitectura escalable de despliegue de servidores web elaborados en el lenguaje de programación PHP y que utilicen el gestor de base de datos Postgresql, para lograr niveles elevados de concurrencia de usuarios, utilizando métodos y herramientas modernas para la solución. Se presentan los resultados alcanzados a un portal desarrollado en el CMS Drupal con la herramienta Apache JMeter, donde se evalúa el consumo de recursos de hardware utilizados y cantidad de peticiones respondidas.

Palabras claves: Sitios web, rendimiento, posicionamiento, despliegue.

ABSTRACT

The web environment becomes a little more complex and competitive each time. An important aspect at positioning of a website is constituted by the performance to satisfy adequately users demands. That is why it is necessary to adopt mechanisms to optimize the deployment of websites by using techniques and tools of good effectiveness. This paper intends to make a suggestion of Free Software for a scalable deployment architecture of web servers developed in the programming language PHP and using the PostgreSQL database manager data to achieve high levels of concurrent of users, using modern methods and tools for the solution. The results achieved to a portal developed in Drupal CMS with Apache JMeter tool, where the consumption of hardware resources used and number of requests answered evaluated are presented.

Keywords: Websites, performance, positioning, deployment.

1. INTRODUCCIÓN

El rendimiento es un elemento importante en el posicionamiento eficaz de los sitios web. La posibilidad de que se carguen rápido las páginas, exista estabilidad en el contenido mostrado y se soporte gran concurrencia de usuarios, influye en que la experiencia del visitante se torne más agradable y productiva. Según un estudio de Forrester Consulting, los usuarios se tornan impacientes cuando las páginas tardan más de 2 segundos en cargar, lo que provoca la insatisfacción y abandono del sitio web visitado (Consulting Forrester, 2009).

En la actualidad existen muchas alternativas para lograr la eficiencia y estabilidad de sitios web desde el punto de vista de software. Entre ellas se encuentran los balanceadores de carga, los aceleradores de código y los sistemas para lograr alta disponibilidad. Si se utilizan correctamente y se dispone de servidores con prestaciones de hardware aceptables, se obtendrá un servicio exquisito.

En este trabajo se pretende realizar una propuesta de arquitectura de despliegue de sitios web desarrollados en PHP para lograr altos niveles de concurrencia y disponibilidad. Se incluye en la propuesta igualmente, un enfoque

orientado a la utilización de Postgresql como gestor de base de datos. Se presentan los resultados obtenidos a un portal desarrollado con el CMS Drupal mediante la herramienta Apache JMeter para evaluar la propuesta realizada.

2. DESARROLLO

Existen variadas técnicas, procedimientos y herramientas que permiten la optimización y estabilidad del funcionamiento de los sitios web. Se perderían usuarios si ellos no están conforme con la velocidad de respuesta que brinda un sitio web (Davison, 2001). Se hace necesario aumentar la capacidad de cómputo para lograr altos niveles de concurrencia de los visitantes y respuestas satisfactorias por parte de los servidores. Algunas técnicas para disminuir los tiempos de respuesta de los servidores web incluyen la utilización de clustering (Iyengar, et al., 2000), (Cardellini, et al., 1999), con lo cual se logra elevar el rendimiento total para acometer tareas que individualmente o en sumatoria simple no podrían realizar; utilización de arquitecturas de despliegue en varios niveles, uso del archivado con caché para la introducción de mecanismos eficientes de distribución de objetos en la Web (Bakhtiyari, 2012), optimización en cuanto a rendimiento de las herramientas de despliegue seleccionadas y mejoramiento del hardware de los servidores en cuanto a memoria y procesamiento fundamentalmente.

De igual forma, lograr la efectividad de alta disponibilidad del servicio brindado por un sitio web en todas las capas de su implementación es un aspecto de vital importancia, que debe ser tenido en cuenta para un posicionamiento en la Web de forma adecuada.

En las próximas secciones se presentará la propuesta de los diferentes niveles de implementación del despliegue de servidores web de altas prestaciones.

2.1.1 BALANCEADOR DE CARGA

Un balanceador de carga permite distribuir la carga a través de múltiples recursos disponibles. De esta forma, se optimiza la utilización del procesamiento, se disminuyen los tiempos de respuesta y se maximiza la disponibilidad de los recursos. En el caso de los balanceadores de carga para servidores web, representan una alternativa eficaz para afrontar los retos de la alta concurrencia.

Existen variadas técnicas para lograr balanceo de carga en servidores web (Bryhni, et al., 2000), (Teo, et al., 2001), todas con sus características particulares, ventajas y desventajas en determinados contextos, pero siempre obteniendo el resultado deseado. Estas técnicas son implementadas de manera exitosa en herramientas de probada efectividad. Una de ellas es HAProxy (Tarreau, 2014), la cual tiene entre sus virtudes ser Software Libre, muy rápida, análisis de logs depurado, autenticación mediante el protocolo HTTP, posibilidad de analizar estadísticas y monitorización mediante una interfaz web para ver su estado. Según sus creadores, esta herramienta es ideal para sitios web con altos niveles de carga.

2.1.2 SERVIDOR DE CACHÉ DE CONTENIDOS ESTÁTICOS

Un servidor de caché permite solicitar a un servidor web real contenido (imágenes, vídeos, scripts, etc.) para almacenarlo y ser servido en futuras peticiones. De esta forma, se evita realizar llamadas al servidor real, acelerando la velocidad de respuesta de las peticiones.

Los servidores de caché más populares hoy día son Apache Traffic Server, Varnish y Squid (Bakhtiyari, 2012). Este trabajo se enfoca en la utilización de Varnish (Varnish Software, 2014) como servidor de caché por su rendimiento y flexibilidad. El mismo posee su propio lenguaje de configuración vcl (Varnish configuration language), el cual brinda un alto nivel de configuración y creación de políticas por parte de los usuarios.

2.1.3 SERVIDOR WEB

Un servidor web es un programa de computadora que usa el protocolo HTTP para enviar páginas web a las computadoras de los clientes cuando el mismo lo solicita (Merriam-Webster, 2014). Hoy día existen muchas alternativas en el mercado, pero es indudablemente Apache HTTP Server (The Apache Software Foundation,

2014), un proyecto consolidado y robusto. Apache brinda seguridad y eficiencia, ha sido el servidor web en Internet más popular desde abril de 1996.

El servidor Apache por defecto no ofrece niveles elevados de respuesta a solicitudes, debe ser configurado en ese sentido (Liu, et al., 2003), lo cual lo convierte en un servidor de gran rendimiento. Su integración con Varnish es excelente (Bakhtiyari, 2012).

2.1.4 SERVIDOR DE BASES DE DATOS

Un servidor de bases permite realizar operaciones de almacenamiento, recuperación, adición, eliminación y modificación de datos sobre bases de datos.

Los sitios web almacenan normalmente su información en bases de datos, las que en ocasiones llegan a ser inmensas y su gestión eficiente se torna cada vez más importante.

En este trabajo se utiliza PostgreSQL (The PostgreSQL Global Development Group, 2014) como gestor de bases de datos. Es un sistema objeto-relacional de bases de datos Open Source poderoso. Lleva en el mercado más de 15 años de actividad, lo cual ha permitido que haya ganado en reputación en la competencia. De forma simultánea, aunque PostgreSQL maneja el rendimiento y concurrencia de forma excepcional mediante la modificación de sus archivos de configuración, se utiliza Pgpool-II (Pgpool Global Development Group, 2011) como sistema para la replicación, balanceo de carga y alta disponibilidad de bases de datos, el cual es un proyecto enfocado en servir de intermediario entre servidores PostgreSQL para aumentar su rendimiento.

2.1.5 ARQUITECTURA PROPUESTA

A continuación se presenta la propuesta de arquitectura de despliegue de servidores web de altas prestaciones, teniendo en cuenta los elementos abordados anteriormente y otros que serán explicados:

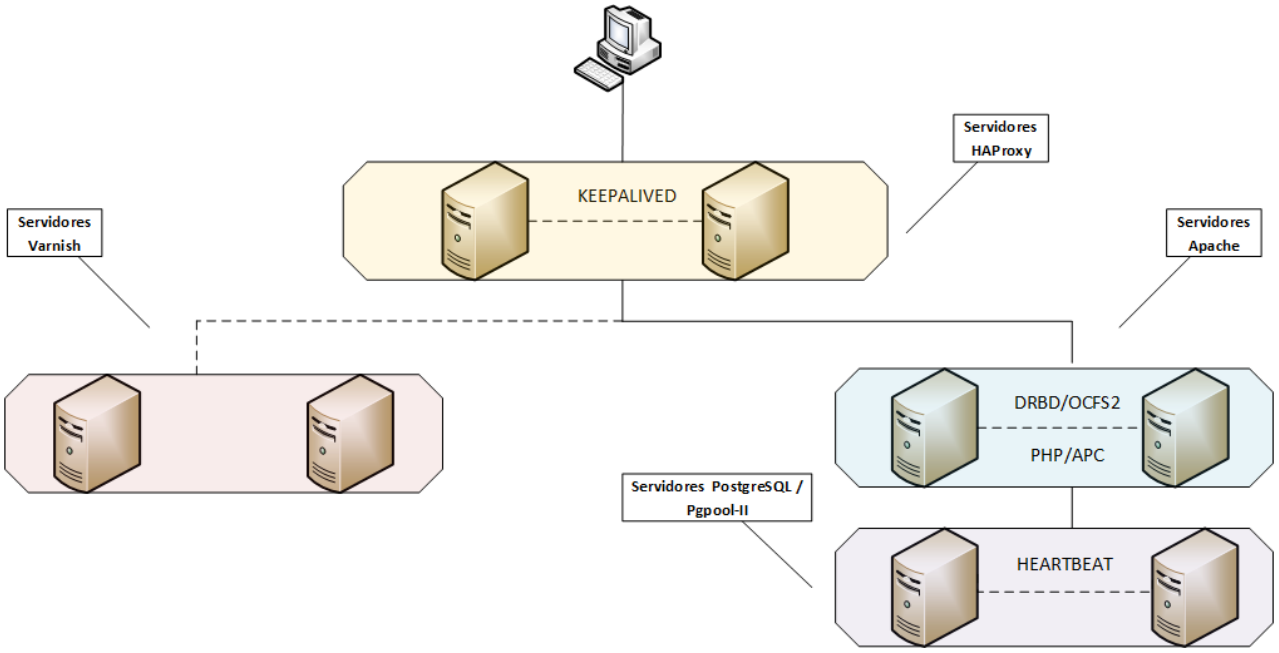

Figura 1 Arquitectura de despliegue de servidores web de altas prestaciones

En la figura 1 se constatan las diferentes capas de las que se compone la propuesta, las que permiten su escalabilidad a más servidores para lograr más potencia de procesamiento.

Cuando un cliente accede al servicio ofrecido por un sitio web, se conectará a un clúster activo-pasivo de HAProxy mediante Keepalived (Keepalived, 2014). Con esto se logra trasladar la dirección IP del servidor activo

al pasivo cuando el primero deje de estar disponible, logrando un nivel de alta disponibilidad robusto. Debido a que estos servidores serán los más expuestos a ataques desde la red (como ataques de Denegación de Servicio), se hace necesario el establecimiento de políticas de seguridad que aseguren su estabilidad en el tiempo. HAProxy puede proteger contra los indeseables ataques de Denegación de Servicio (Tarreau, 2014).

Basado en la URL o extensión del fichero que es requerido por el cliente HAProxy, el mismo detectará si es contenido estático, el cual será enviado a algún servidor Varnish, los que están representados en un clúster activo-activo. Si este servidor tiene el objeto en caché, lo redirigirá al servidor HAProxy para que sea entregado al cliente. En caso de contenido dinámico, HAProxy elegirá un servidor que atienda la petición, el cual enviará su respuesta a HAProxy para atender al cliente.

El lenguaje de programación del lado del servidor utilizado es PHP (The PHP Group, 2014). Es por ello que se utiliza el acelerador de código libre APC (Alternative PHP Cache), el cual le impregna al código del lado del servidor gran velocidad de ejecución (Padilla and Hawkins, 2011).

Por otro lado, el clúster de servidores web debe estar sincronizado en cuanto a la información que se almacena en ellos, utilizándose la combinación DRBD (Guimaraes, 2008) y el sistema de ficheros OCFS2 (Oracle Open Source, 2009) para la replicación de datos entre los nodos. DRBD permite construir un bloque de clústeres de alta disponibilidad mediante un espejo de dispositivos en red. Con OCFS2 se logran escrituras concurrentes en los servidores sin que se corrompa la información. La partición donde se escribirán los datos debe ser de acuerdo a los requerimientos del sitio web; para permitir reajustes de la misma se propone el uso de LVM (CentOS project, 2014). De esta forma, cuando se escriben datos en alguno de los servidores del clúster web, se estará almacenando de igual forma en el resto de una forma segura y rápida.

Por último, los servidores web para consumir los servicios de bases de datos, se conectan a una dirección IP flotante generada por Heartbeat (Linux-HA, 2010) de un clúster de alta disponibilidad de pgpool-II.

Pgpool-II maneja todas las conexiones que se realizan a los servidores PostgreSQL, permitiendo la replicación, balanceo de carga, límite de conexiones excedentes y ejecución de consultas paralelas en dichos servidores. De esta forma, se obtiene un alto rendimiento y estabilidad del servicio de bases de datos.

3. RESULTADOS

Se realizaron pruebas de carga y estrés con la herramienta Apache JMeter (Apache Software Foundation, 2014) a un portal desarrollado en el CMS Drupal (Buytaert, 2014). Se utilizaron las herramientas y servidores que fueron descritos en la arquitectura propuesta. Los servidores fueron virtualizados en un clúster de virtualización en Proxmox (Proxmox Server Solutions GmbH, 2014) compuesto por 2 servidores físicos. Las características de los servidores son:

- Servidores físicos: Debian 7, 8 GB RAM, 4 núcleos, Core i3-2100
- Servidores virtuales HAProxy: Debian 7, 1 GB RAM, 2 cores.
- Servidores virtuales Varnish: Debian 7, 2 GB RAM, 2 cores.
- Servidores virtuales web: Debian 7, 1 GB RAM, 2 cores.
- Servidores virtuales de bases de datos: Debian 7, 1 GB RAM, 2 cores.

Las métricas utilizadas se enfocaron en la medición del tiempo promedio de respuesta a las peticiones y el consumo de recursos, principalmente el porciento de utilización de CPU y RAM. Se realizaron 3 pruebas, simulando diferentes cantidades de usuarios concurrentes a diferentes partes críticas del portal (página principal [PP], galería de imágenes [GI], administración del sitio [AS]). La medición de los recursos de hardware al servidor físico se realizó con la herramienta htop (Muhammad, 2014). A continuación se presentan los resultados obtenidos:

- Primera prueba *(20 usuarios concurrentes, 50 ciclos)*:

Tabla 1: Primera prueba realizada con 1000 muestras

	Muestras	Prom. de tiempo de respuesta (ms)	Tiempo mínimo de respuesta (ms)	Tiempo máximo de respuesta (ms)	Error %	Throughput (pág/seg)	Throughput (Kb/seg)
PP	1000	344	306	414	0	23,15	865,89
GI	1000	267	255	273	0	24,18	658,75
AS	1000	142	133	155	0	28,15	898,76
Totales	3000	251	133	414	0	75,48	2423,40

El promedio de consumo físico de recursos para esta prueba fue de 1 GB de memoria RAM y 25 % de utilización de los núcleos del procesador.

- Segunda prueba *(100 usuarios concurrentes, 50 ciclos)*:

Tabla 2: Segunda prueba realizada con 5000 muestras

	Muestras	Prom. de tiempo de respuesta (ms)	Tiempo mínimo de respuesta (ms)	Tiempo máximo de respuesta (ms)	Error %	Throughput (pág/seg)	Throughput (Kb/seg)
PP	5000	434	396	453	0	56,45	1854,54
GI	5000	410	384	435	0	56,40	1956,33
AS	5000	390	365	412	0	58,33	1845,44
Totales	15000	411	365	412	0	171,18	5656,31

El promedio de consumo físico de recursos para esta prueba fue de 4 GB de memoria RAM y 55 % de utilización de los núcleos del procesador.

- Tercera prueba *(200 usuarios concurrentes, 60 ciclos)*:

Tabla 3: Tercera prueba realizada con 12000 muestras

	Muestras	Prom. de tiempo de respuesta (ms)	Tiempo mínimo de respuesta (ms)	Tiempo máximo de respuesta (ms)	Error %	Throughput (pág/seg)	Throughput (Kb/seg)
PP	12000	675	610	791	0	83,78	2054,54
GI	12000	632	596	799	0	82,54	1987,58
AS	12000	598	588	659	0	89,69	2103,69
Totales	36000	635	588	799	0	256,01	6145,81

El promedio de consumo físico de recursos para esta prueba fue de 6 GB de memoria RAM y 75 % de utilización de los núcleos del procesador.

Como se puede constatar, a medida que se aumentó el número de iteraciones, aumentó el número de recursos consumidos pero en sentido general los valores obtenidos son satisfactorios teniendo en cuenta la cantidad de muestras procesadas.

La página de administración fue la que mejores resultados presentó, aunque teniendo en cuenta que la página principal era la más cargada, sus resultados fueron importantes. Los servidores de caché Varnish influyeron en estos resultados teniendo en cuenta que la mayoría del contenido era estático.

4. CONCLUSIONES

Se ha presentado una propuesta general de Software Libre para la implementación de servidores web de altas prestaciones. La solución puede ser adaptada a contextos específicos de acuerdo a las necesidades detectadas. De igual forma, la propuesta permite adaptar otras herramientas informáticas que ofrezcan niveles de rendimiento similares a los obtenidos.

Las pruebas realizadas, aunque no se hicieron en un entorno real, son un punto de partida significativo para evaluar la perspectiva de la solución en entornos más complejos. Los resultados obtenidos son satisfactorios si se tiene en cuenta el hardware utilizado y las muestras procesadas.

REFERENCIAS

Bakhtiyari, Shahab. (2012). "Performance Evaluation of the Apache Traffic Server and Varnish Reverse Proxies."

Buytaert, D. (2014). "Drupal", https://drupal.org/, 20/01/2014.

Bryhni, Haakon, Espen Klovning, and Oivind Kure. (2000). "A comparison of load balancing techniques for scalable web servers." Network, IEEE 14.4: 58-64.

Cardellini, Valeria, Michele Colajanni, and Philip S. Yu. (1999). "Dynamic load balancing on web-server systems." Internet Computing, IEEE 3.3: 28-39.

CentOS Project. (2014). "LVM Administrator's Guide". http://www.centos.org/docs/5/html/Cluster_Logical_Volume_Manager/, 22/01/2014.

Consulting, Forrester. (2009). "eCommerce Web site performance today: an updated look at consumer reaction to a poor online shopping experience." White Paper of Akamai Technologies Inc.

Davison, B.D. (2001). "A web caching primer". Internet Computing, IEEE, 5(4):38–45.

Guimaraes, Luciana. (2008). "Confidentiality, integrity and high availability with open source IT green." arXiv preprint arXiv:0805.4394.

Iyengar, Arun, et al. (2000). "High performance web site design techniques." Internet Computing, IEEE 4.2: 17-26.

Keepalived. (2014). "Load balancing & High Availability", http://www.keepalived.org/pdf/asimon-jres-paper.pdf, 15/01/2014.

Linux-HA. (2010). "Heartbeat", http://linux-ha.org/wiki/Heartbeat, 20/01/2014.

Liu, Xue, et al. (2003). "Online response time optimization of apache web server." Quality of Service—IWQoS 2003. Springer Berlin Heidelberg, 461-478.

Merriam-Webster. (2014). "Server definition", http://www.merriam-webster.com/dictionary/server, 20/01/2014.

Muhammad, Hisham. (2014). "htop - an interactive process viewer for Linux", http://hisham.hm/htop/index.php?page=main, 25/01/2014.

Oracle Open Source. (2009). "OCFS2", https://oss.oracle.com/projects/ocfs2/, 18/01/2014.

Padilla, Armando, and Tim Hawkins. (2011). "Opcode Caching." Pro PHP Application Performance. Apress 83-107.

Pgpool Global Development Group. (2011). "PGPool-II", http://pgpool.projects.pgfoundry.org/pgpool-II/doc/pgpool-en.html, 20/01/2014.

Proxmox Server Solutions GmbH. (2014). Proxmox, http://www.proxmox.com/, 20/01/2014.

Tarreau, W. (2014). "HAProxy. The Reliable, High Performance TCP/HTTP Load Balancer." from http://haproxy.1wt.eu/, 20/01/2014.

Teo, Yong Meng, and Rassul Ayani. (2001). "Comparison of load balancing strategies on cluster-based web servers." Simulation 77.5-6: 185-195.

The Apache Software Foundation. (2014). "Apache HTTP Server", http://httpd.apache.org/, 20/01/2014.

The PostgreSQL Global Development Group. (2014). "PostgreSQL", http://www.postgresql.org, 20/01/2014.

Varnish Software. (2014). "Varnish", https://www.varnish-cache.org/, 20/01/2014.

Authorization and Disclaimer

Authors authorize LACCEI to publish the paper in the conference proceedings. Neither LACCEI nor the editors are responsible either for the content or for the implications of what is expressed in the paper.