BALANCEO DE CARGA DE SERVIDORES CON MOD_PROXY_BALANCER

Cristian David Agredo: Juan Esteban Alarcón; Juan Diego Caicedo; Esteban Sardi

Facultad de Ingeniería, Departamento de informática
Universidad Autónoma de Occidente
Cali, Colombia

cristian.agredo@uao.edu.co; juan_esteban.alarcon@uao.edu.co; juan_diego.caicedo@uao.edu.co; esteban.sardi@uao.edu.co

Abstract — In this project, the implementation of a cluster of web servers with load balancing will be carried out, this will work as frontend of the web server and each time a request is sent to the load balancer and it will be in charge of redirecting the request to one of the servers of the cluster hosted on the backed, these will have the necessary resources to resolve the requests.

The web requests are made to the balancer, which will be in charge of delegating the server in charge of processing the request, the backed servers will be running a web service. Finally, the tests were carried out implementing different load balancing algorithms to analyze the results obtained and see how each of these solves the requests made.

I. INTRODUCCIÓN

Con la tendencia actual de los entornos web, y la gran cantidad de personas que necesitan acceder a ellos de forma rápida y en cualquier momento el utilizar un balanceador de carga se volvió una práctica esencial si se desea que una página web con una alta demanda de usuarios pueda siempre mantenerse disponible para responder a todas las peticiones que se realizan sobre ella.

En esta práctica se explica la manera de crear un balanceador de carga local usando apache mod proxy balancer desde el sistema operativo de CentOS 7, como hacer las distintas pruebas de carga con Jmeter y analizarlas para poder seleccionar el algoritmo de balanceo que mejor se adapte a nuestra página.

Para proceder con la práctica es necesario tener un conocimiento básico en sistemas operativos tipo GNU/Linux, se recomienda también usar una box en Vagrant con el fin de optimizar los recursos de la máquina en la que se proceda con la práctica.

II. OBJETIVOS

A. General

Implementar un clúster de servidores web con balanceo de carga, el cual funcionará como frontend y se encargará de redirigir las peticiones a uno de los servidores alojados en el backend para balancear la carga de peticiones.

B. Específicos

- Realizar la configuración de los servidores requeridos para el desarrollo del proyecto y su balanceador
- Probar la ejecución de múltiples peticiones simultáneas al cluster de servidores web
- Realizar pruebas de carga variando los parámetros de las mismas
- Obtener los resultados y conclusiones de los diferentes algoritmos de balanceo de carga

II. MARCO TEÓRICO

A. Cluster/Balanceador de carga

Consiste en distribuir de forma eficiente el tráfico de red que reciben los sitios web por medio de los servidores backend, puesto que en la actualidad los sitios web modernos de alto tráfico se ven en la obligación de atender millones de peticiones simultáneas de todos sus usuarios ya sea para acceder a videos, texto, datos, etc. Todo esto de la forma más rápida y confiable posible, se puede apreciar como funciona de forma gráfica en la figura 1.

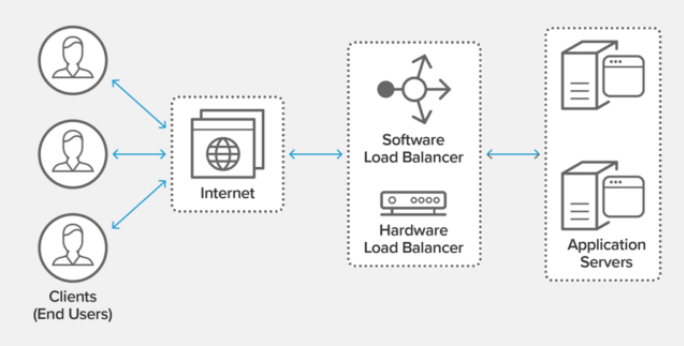


Figura 1. Diagrama de un balanceador de carga

El balanceador actúa como un policía de tránsito entre los servidores y las peticiones entrantes para enrutarlas con el servidor mejor capacitado para atender la solicitud, de esta forma se evita que algún servidor reciba más peticiones de las que puede atender y se empiece a saturar reduciendo su rendimiento y el de todo el entorno.

B. Apache mod_proxy_balancer:

Es el módulo de apache que nos proveé el servicio de balanceo de carga para los protocolos soportados como HTTP, FTP, AJP13, WebSocket. Con el fin de no tener que

realizarlo desde 0 y que no sea tan complejo crear e implementar el balanceador de carga en los servidores.

C. Máquina virtual

Consiste en un entorno virtual que nos permite utilizar un sistema operativo aprovechando y particionando los recursos de memoria, disco, etc, de la máquina local.

Esto es posible gracias a la tecnología de la virtualización que utiliza el software para simular el hardware de la máquina virtual lo que permite según la capacidad de la máquina local correr una o varias máquinas virtuales al mismo tiempo gracias a un hipervisor que es el encargado de virtualizar los recursos locales y repartirlos en las máquinas virtuales que se necesiten.

D. Apache

Es el software de servidor web más utilizado en la actualidad. Desarrollado y mantenido por Apache Software Foundation, apache es un software de código abierto y gratuito, es ejecutado en el 67% de los servidores web del mundo. Su gran capacidad de personalización gracias a los diferentes entornos y módulos que tiene le permite satisfacer las necesidades de muchas compañías como por ejemplo WordPress.

.

E. Servidor Web

El término servidor web puede referirse a software o hardware, o ambos trabajan conjuntamente:

Por el lado del hardware, un servidor web es una computadora que almacena un software de servidor web y unos archivos que componen un sitio web (por ejemplo, documentos HTML, imágenes, hojas de estilo CSS y archivos JavaScript). Un servidor web se conecta a internet y soporta intercambio físico de datos con otros dispositivos conectados a la web.

Por el lado del software, un servidor web incluye varias partes que controlan cómo los usuarios de la web acceden a los archivos alojados. Como mínimo, este es un servidor HTTP. Un servidor HTTP es un software que entiende URLs (direcciones web) y HTTP (el protocolo que el navegador usa para ver páginas web). Un servidor HTTP puede ser accedido a través de nombres de dominio de los sitios web que lo almacena, y este entrega el contenido a estos sitios web alojados hacia el dispositivo del usuario final.

F. Protocolo HTTP

Es un protocolo para extraer recursos en formato de documentos HTML. Es la base de cualquier intercambio de datos en la web y es un protocolo cliente-servidor, lo que significa que las solicitudes son iniciadas por el destinatario, que usualmente es un navegador web. Para completar la petición, un documento completo es reconstruido desde diferentes subdocumentos extraídos, tales como texto, descripción de capas, imágenes, videos, scripts y demás.

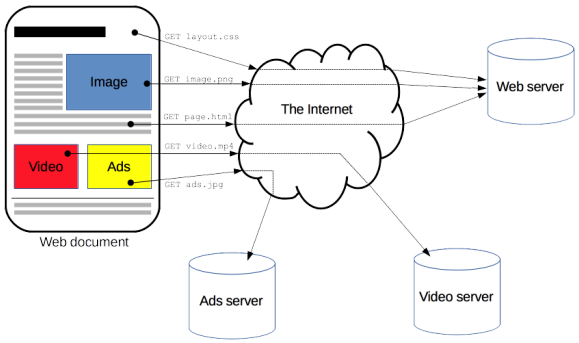


Figura 2. Proceso de creación de documento HTML.

Clientes y servidores se comunican intercambiando mensajes individuales (a diferencia de un flujo de datos). Los mensajes son envíados por el cliente, usualmente un navegador web, son llamados *"request"* y los mensajes son enviados por el servidor como respuesta y son llamados *"responses"*.

III. MATERIALES Y MÉTODOS

Para el desarrollo del proyecto se requirió de un PC (Equipo de cómputo personal), un software de virtualización (VirtualBox), Máquinas virtuales con el sistema operativo CentOS 7.9 y una herramienta para generar entornos de desarrollo (Vagrant).

A. PC (Equipo de cómputo personal)

El computador es una herramienta que nos facilita mucho las tareas que realizamos en el día a día. Para el desarrollo del proyecto este se ha utilizado para alojar los diferentes servidores configurados, el sistema de virtualización en donde se han instalado los sistemas operativos y hacer uso de la herramienta de vagrant.

Figura 3 . Equipo personal.

Las características del equipo han variado según los integrantes del equipo. Sin embargo, las características recomendadas para realizar el desarrollo del proyecto son 8GB de memoria ram, ya que se requieren diferentes servidores en funcionamiento. Se ha probado con 4GB pero lamentablemente no dio los resultados esperados. Almacenamiento en disco duro de 2GB para las configuraciones realizadas y un procesador compatible con la arquitectura Intel X86 de Intel X86 64.

B. Software de virtualización (VirtualBox)

El software de virtualización de VirtualBox nos ayudará a montar máquinas virtuales con instalaciones de sistemas operativos, en este caso cuatro máquinas virtuales (servidores) con el sistema operativo de CentOS 7.9. Nos permite probar aplicaciones o servicios en otro sistema operativo sin la necesidad de realizar particiones e instalarlos junto a nuestro sistema operativo principal.

Figura 4.Logo de VirtualBox.

Instalar un sistema operativo en una máquina virtual puede ser muy útil a la hora de realizar configuraciones más profundas en el sistema operativo o instalar alguna aplicación peligrosa que puede hacer que tu máquina principal falle.

C. Sistema operativo CentOS 7.9

Este sistema operativo es una distribución de linux apoyado en RHEL (Red Hat Enterprise Linux) su uso principal está destinado para el sector empresarial y organizaciones de gran tamaño.

Figura 5. Logo de CentOS.

La razones por las que se ha escogido este sistema operativo es porque los requerimientos de hardware para su correcto funcionamiento son bastante accesibles, además de que está siendo implementado en un sistema de virtualización, este funciona asignándole 512 MB de memoria ram. Sin embargo para una mayor fluidez en el desarrollo del proyecto se le asignó 1024MB de memoria ram. debe tener 20GB de espacio en disco duro o un poco menos si se usa desde línea de comandos. Toda la práctica se llevó a cabo sin interfaz gráfica del sistema operativo, todo a través de línea de comandos desde el Vagrant

D. Vagrant

Vagrant es una herramienta que nos ayuda a crear y manejar máquinas virtuales con un mismo entorno de trabajo, este como tal no tiene la capacidad para correr una máquina virtual, se encarga de las características con las que se debe crear esta máquina y los complementos a instalar.

Figura 6. Logo de Vagrant.

Sabemos que puede ser muy tedioso cuando debemos desarrollar un proyecto en equipo desde diferentes máquinas y la configuración que realizamos no es la misma de todos los integrantes o simplemente por otras configuraciones ajenas no funciona de la manera en como se esperaba. Desde el desarrollo del proyecto se utilizaron funciones de vagrant como el "Vagrant Cloud" en donde se ha montado la configuración final de las máquinas para que el colectivo no presente los problemas anteriormente mencionados.
La gran ventaja de esto es que posee un archivo de configuración llamado Vagrantfile donde se centraliza toda la configuración que queremos asignarle a la máquina virtual.

IV. METODOLOGÍA

Este proyecto consta de implementar tres tipos de servidores, uno de ellos tiene la función de ser el balanceador (Frontend) de carga, el cual es el que delega cual servidor resuelve las peticiones realizadas por el cliente, y otros dos, encargados de correr un servicio web en el Backend. Sin embargo, existen diferentes algoritmos de balanceo de carga para resolver las peticiones los cuales implementamos dos de los más utilizados que serán byrequest (equilibrado por el número de peticiones) y bytraffic (balanceado de acuerdo con el tráfico).Por lo tanto para realizar las pruebas y hacer una análisis más exhaustivo se ha configurado un cuarto servidor que funciona como balanceador, alojando uno de los algoritmos anteriormente mencionado.

En el primer servidor, el cual se ha llamado como balanceador1, está alojado el cluster de balanceo de carga asignado con la ip 192.168.50.30, este es el encargado de redirigir las peticiones a los servidores del backed con el servicio web. Este balanceo está realizado mediante el algoritmo byrequest (equilibrado por el número de peticiones) y mejor llamado como Round Robin

En el segundo servidor, el cual se ha llamado como balanceador2, está alojando un cluster de balanceo de carga con la ip 192.168.50.40, este es el encargado de redirigir las peticiones a los servidores del backend con el servicio web. Este balanceo está realizado mediante el algoritmo bytraffic (balanceado de acuerdo al tráfico).

Finalmente para los otros dos servidores, estos son los encargados de implementar un servicio web en el backend con el lenguaje de programación html. estos fueron llamados como maquina1 y maquina2, con las ip's 192.168.50.10 y 192.168.50.20 respectivamente.

Los integrantes del colectivo se han delegado el trabajo de manera equitativa realizando cada uno la configuración de los servidores para realizar pruebas y analizar resultados y errores encontrados durante el proceso

V. INSTALACION Y CONFIGURACION

Configuración del balanceador 1 y 2. En la configuración explicamos las variaciones respecto al segundo balanceador, ya que solo se requiere cambiar el algoritmo con el que se resuelven las peticiones

```
//Nos logueamos como usuario root
#sudo -i
//Instalamos el editor de texto Vim
#yum install vim -y
//Instalamos el servicio http
#yum install httpd httpd-manual -y
//iniciamos el servicio
#systemctl start httpd
//Habilitamos el servicio para que se inicie cuando encienda la máquina
#systemctl enable httpd
//Verificamos el puerto en el que escucha httpd
#grep -i listen /etc/httpd/conf/httpd.conf
```

```
[root@balanceador ~]# grep -i listen /etc/httpd/conf/httpd.conf
# Listen: Allows you to bind Apache to specific IP addresses and/or
# Change this to Listen on specific IP addresses as shown below to
#Listen 12.34.56.78:80
Listen 80
```

```
//Verificamos que esté instalado el módulo de proxy_balancer_module
#httpd -M | grep proxy
```

```
[root@balanceador ~]# httpd -M | grep proxy
AH00558: httpd: Could not reliably determine the serv
ame' directive globally to suppress this message
 proxy_module (shared)
 proxy_ajp_module (shared)
 proxy_balancer_module (shared)
 proxy_connect_module (shared)
 proxy_express_module (shared)
 proxy_fcgi_module (shared)
 proxy_fdpass_module (shared)
 proxy_ftp_module (shared)
 proxy_http_module (shared)
 proxy_scgi_module (shared)
 proxy_wstunnel_module (shared)
```

```
//verificamos que el algoritmo por el que resuelve las peticiones esté activo byrequest
##httpd -M | grep requests
```

```
[root@balanceador ~]# httpd -M | grep requests
AH00558: httpd: Could not reliably determine the
ame' directive globally to suppress this message
 lbmethod_byrequests_module (shared)
[root@balanceador ~]#
```

```
//Activamos el servicio del firewall
#service firewalld start
//Agregamos al firewall el servicio http
#firewall-cmd --permanent --add-service=http
//Reiniciamos el firewall
#firewall-cmd --reload
//El servicio de http por defecto trae una página principal, eliminamos esta para agregar una personalizada
#rm /etc/httpd/conf.d/welcome.conf
//Reiniciamos el servicio de http
#systemctl restart httpd
//Vamos a la siguiente ruta de configuración de http
#cd /etc/httpd/conf.d/
//Creamos y editamos el archivo vhost_default_00.conf
#vim vhost_default_00.conf
//Dentro de este archivo colocamos el siguiente código

<VirtualHost *:80>
        ProxyRequests off
        <Location /balancer-manager>
                SetHandler balancer-manager
                Require all granted
         </Location>
        <Proxy balancer://cluster1>
                balancerMember http://192.168.50.10:80
                balancerMember http://192.168.50.20:80
              ProxySet lbmethod=byrequests //si se requiere equilibrar por número de peticiones
#             ProxySet lbmethod=bytraffic //Si se requiere resolver las peticiones por tráfico
        </Proxy>
         ProxyPreserveHost On
        ProxyPass /balancer-manager !
        ProxyPass / balancer://cluster1
         ProxyPassReverse / balancer://cluster1
</VirtualHost>
//Guardamos el archivo con ctrl + c y :wq
//Reiniciamos el servicio http
#systemctl restart httpd

//En la configuración explicamos la variación respecto a los balanceadores, el balanceador
```

```
resuelve las peticiones manteniendo el equilibro
por número de peticiones y el balanceador 2
resuelve las peticiones respecto al tráfico
```

Instalación y configuración de las máquinas 1 y 2. La configuración para estos dos servidores es igual, pues son los que alojan el servicio web. Sin embargo, estos servicios web son diferentes pero se pueden configurar a gusto de cada desarrollador.

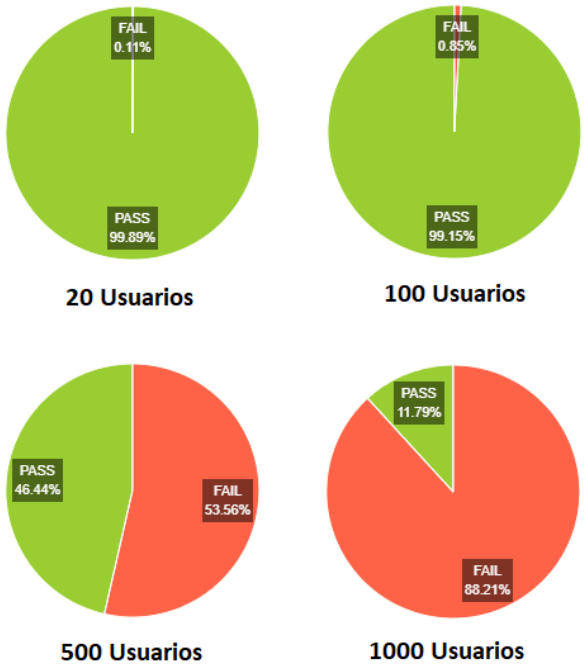
```
//Ingresamos como usuario root
#sudo -i
//Instalamos el editor de texto Vim
#yum install vim -y
//Instalamos el servicio http
#yum install httpd -y
//Nos dirigimos a la siguiente ruta
#cd /var/www/html/
//Creamos y editamos el index del servicio web
#vim index.html
//Colocamos el código dentro del archivo
//Guardamos el archivo ctrl + c y :wq
//Reiniciamos el servicio http
#systemctl httpd restart
```

VI. ANÁLISIS Y RESULTADOS

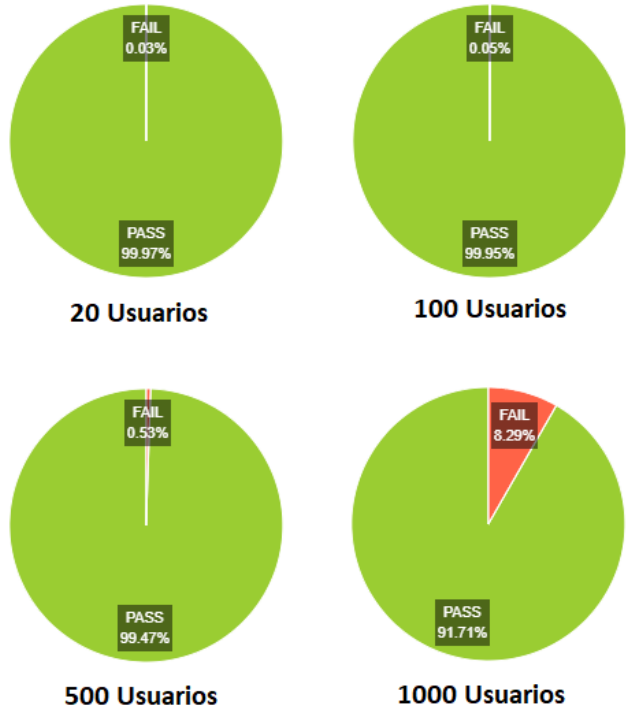
Luego de realizar la instalación de los diferentes servidores, mediante el software de JMeter se han analizado los resultados obtenidos a partir de los algoritmos de balanceo de Round Robin y weighted east-connection. Para cada algoritmo se hicieron pruebas con 20, 100, 500 y 1000 usuarios, durante cada intervalo de 10 segundos se ingresaban un número de usuarios simultáneamente hasta completar el total de usuarios de la prueba por un transcurso total de 6 minutos para cada prueba. Por ejemplo para las pruebas con 1000 usuarios se ingresaban en promedio 28 usuarios cada 10 segundos hasta completar los mil en los 6 minutos.

A continuación se presentarán los resultados obtenidos de las pruebas realizadas para cada uno de los algoritmos mencionados:

Gráfica 1. Resumen de peticiones con algoritmo Round Robin.



Gráfica 2. Resumen de peticiones con algoritmo Weighted east-connection.



En las gráficas 1 y 2 se puede observar el resultado de las pruebas ejecutadas para 20, 100, 500 y 1000 usuarios en donde se puede

mostrar el porcentaje de peticiones exitosas versus las que fallan. Con los resultados obtenidos podemos observar que el algoritmo que mejor responde a las peticiones exitosas de muchos usuarios es el de weighted east-connection, ya que el round robin empieza a tener una tasa de más del 50% de fallos después de los 500 usuarios, ya para los 1000 usuarios la tasa de fallos es mayor al 88% lo cual sería inconcebible para la ejecución de un servidor web que requiera gran cantidad de flujo de usuarios. En contraparte, el weighted east-connection se mantiene por debajo del 0.6% de fallas incluso con 500 usuarios concurrentes, para los 1000 usuarios su tasa de fallos es tan solo del 8.29%, un valor muy bajo considerando los valores arrojados por round robin.

Tabla 1. Porcentajes de tipos de errores generados usando el algoritmo Round Robin para 1000 usuarios.

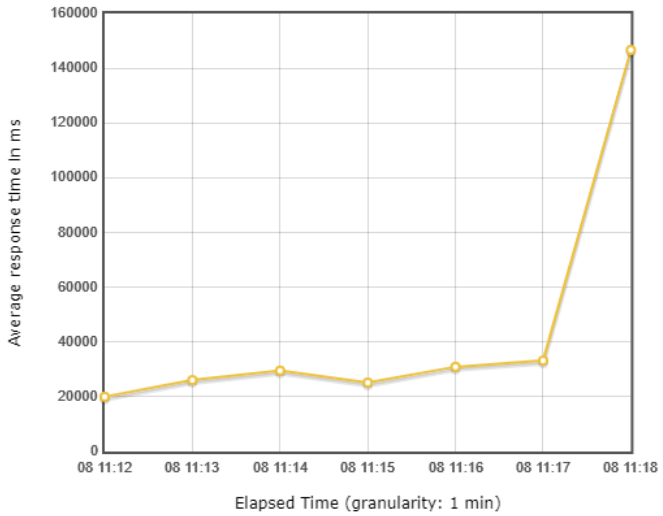
Type of error	Number of errors	% in errors	% in all samples
Non HTTP response code: org.apache.http.conn.HttpHostConnectException/Non HTTP response message: Connect to 192.168.50.30:80 [/192.168.50.30] failed: Connection timed out: connect	5135	77.50%	68.36%
Non HTTP response code: java.net.SocketTimeoutException/Non HTTP response message: Read timed out	1420	21.43%	18.90%
Non HTTP response code: java.net.SocketException/Non HTTP response message: Connection reset	42	0.63%	0.56%
502/Proxy Error	29	0.44%	0.39%

Tabla 2. Porcentajes de tipos de errores generados usando el algoritmo Weighted east-connection para 1000 usuarios.

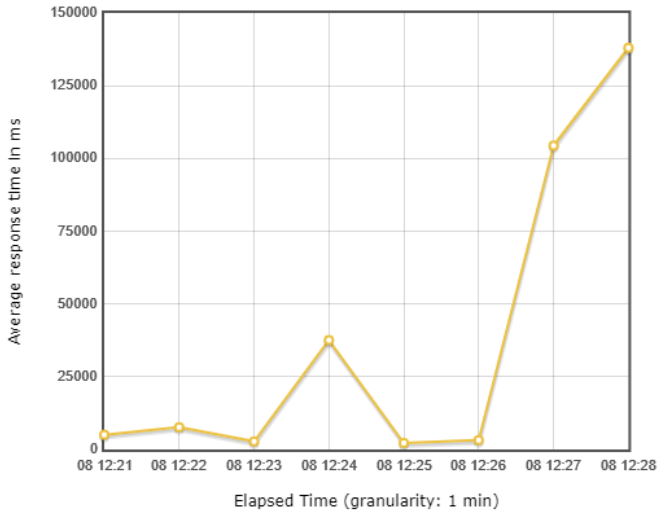
Type of error	Number of errors	% in errors	% in all samples
Non HTTP response code: org.apache.http.conn.HttpHostConnectException/Non HTTP response message: Connect to 192.168.50.40:80 [/192.168.50.40] failed: Connection timed out: connect	3222	70.01%	5.81%
Non HTTP response code: java.net.SocketException/Non HTTP response message: Connection reset	675	14.67%	1.22%
Non HTTP response code: java.net.SocketTimeoutException/Non HTTP response message: Read timed out	589	12.80%	1.06%
Non HTTP response code: org.apache.http.NoHttpResponseException/Non HTTP response message: 192.168.50.40:80 failed to respond	116	2.52%	0.21%

Como se observa en la tabla 1 y 2, los errores comunes arrojados en ambos algoritmos son HttpHostConnectException, SocketTimeoutException y SocketException, de los cuales el más común es el HttpHostConnectException que se genera debido a que un usuario trata de acceder al servidor a través de TCP pero el servidor no dispone en ese momento de ningún listener para aceptar la conexiones solicitada. Podemos observar además que cada algoritmo genera un error único que no maneja el otro, en el caso del Round Robin tenemos un error 502/Proxy Error y en el caso del Weighted east-connection tenemos uno de tipo NoHttpResponseException.

Gráfica 3. Porcentajes de promedio de tiempos de respuesta (en milisegundos) en el periodo de duración de la prueba de 6 minutos usando el algoritmo Round Robin para 1000 usuarios.



Gráfica 4. Porcentajes de promedio de tiempos de respuesta (en milisegundos) en el periodo de duración de la prueba de 6 minutos usando el algoritmo Weighted east-connection para 1000 usuarios.

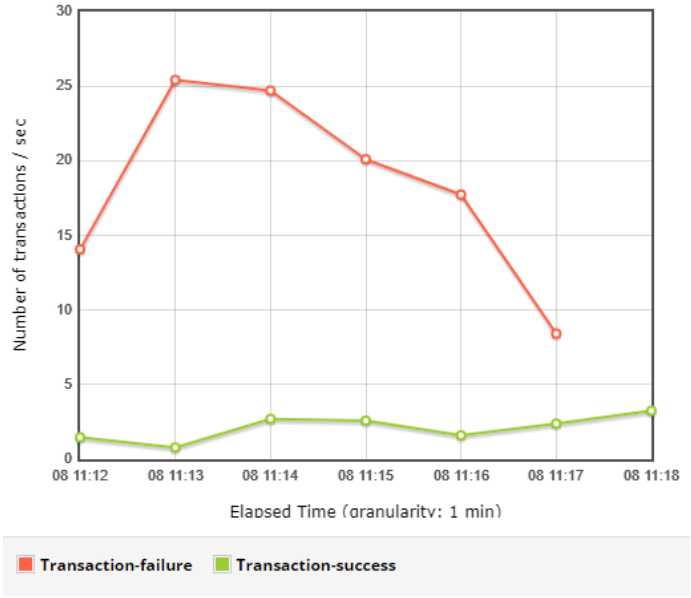


Podemos observar en en ambas gráficas los tiempos de de respuesta en promedio que tarda en responder las solicitudes los balanceadores de ambos algoritmos en los 6 minutos de duración de cada prueba, los resultados se muestran en cada minuto exactamente, razón por la cual la gráfica se ve algo plana, sin embargo es suficiente para obtener algunos análisis los cuales se detallarán a continuación.

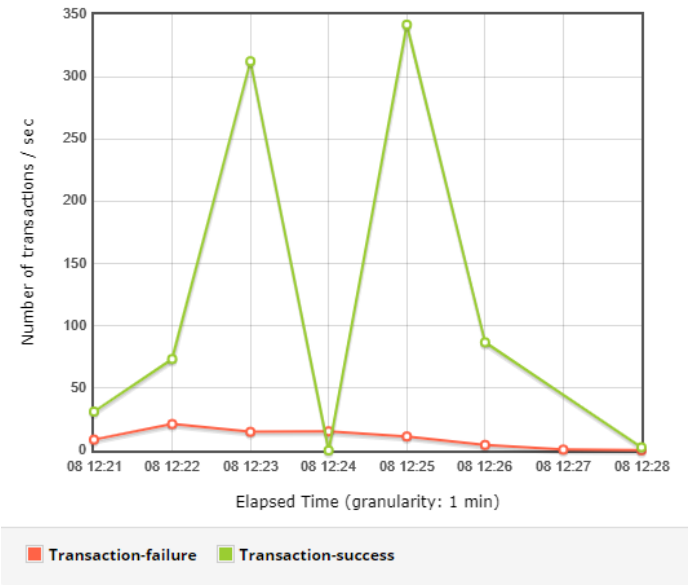
Si observamos los tiempos de respuesta al final de ambos algoritmos, podemos ver que no difieren tanto uno del otro (a diferencia de lo que

se observaba en la gráfica del porcentaje de errores). Sin embargo es notorio que con el algoritmo Weighted east-connection se obtiene un tiempo de respuesta final (cuando ya se han intentado conectar los 1000 usuarios) un poco más corto, que el Round Robin, siendo alrededor de 8500 milisegundos menor que este último. También podemos ver que durante los minutos anteriores los tiempos de respuestas si son considerablemente menores con Weighted east-connection, sin embargo considerar que este algoritmo sube sus tiempos de respuesta considerablemente más que el Round Robin un minuto antes, siendo que su pico de subida comienza en el minuto 4, mientras que en el Round Robin lo hace en el minuto 5, es decir al último minuto de la prueba, cuando está a punto de llegar a los 1000 usuarios.

Gráfica 5. Total de transacciones por segundo en el periodo de duración de la prueba de 6 minutos usando el algoritmo Round Robin para 1000 usuarios.



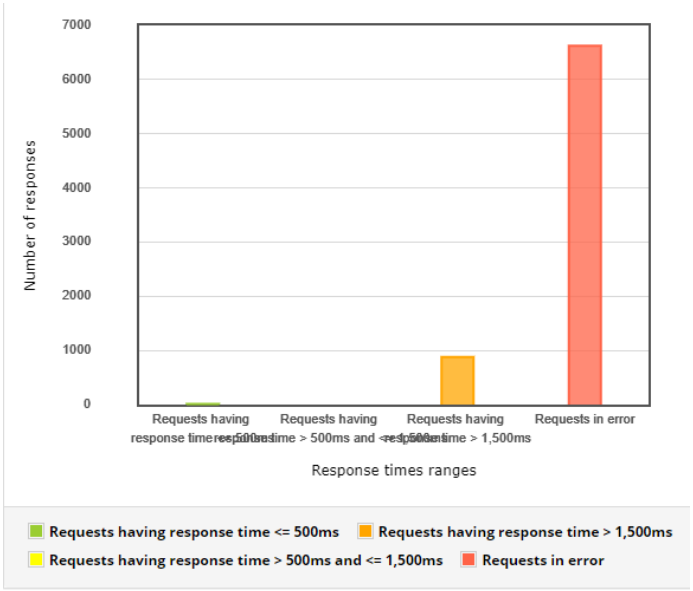
Gráfica 6. Total de transacciones por segundo en el periodo de duración de la prueba de 6 minutos usando el algoritmo Weighted east-connection para 1000 usuarios.



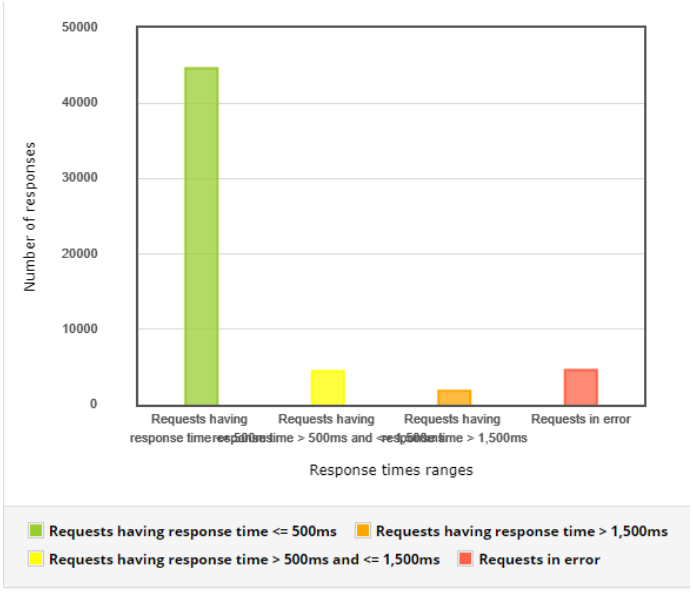
La gráfica 5 y 6 muestra el total de transacciones por cada segundo que son exitosas versus las que fallan en el tiempo de duración de las pruebas que fueron 6 minutos para los balanceadores hechos con los algoritmos Round Robin y Weighted east-connection respectivamente.

Analizando ambas gráficas podemos observar, al igual que lo pudimos hacer en las gráficas 1 y 2 que el Round Robin maneja un número mucho mayor de transacciones fallidas que el Weighted east-connection, sin embargo aquí no se habla de porcentajes sino de número de transacciones a través del tiempo de duración, lo que nos permite evidenciar el comportamiento de los fallos con respecto a los éxitos en cada periodo de tiempo (en este caso en cada minuto). Esto nos permite observar que con Round Robin desde un comienzo se tienen un mayor de porcentajes de fallas que de éxitos, siendo esta diferencia bastante considerable (aprox 14 fallas y 2 exitos), y esta cantidad de fallas sube considerablemente en el siguiente minuto, sin embargo a partir de este los fallos empiezan a disminuir de manera constante hasta no tener ninguno a en el último minuto. La gráfica del balanceador que usa Weighted east-connection muestra un comportamiento muy similar al otro, ambos aumentan sus fallas en el primer minuto y a partir de ahí, estas comienzan a disminuir, sin embargo hay dos diferencias importantes que tiene este último algoritmo: la primera s que el número de transacciones fallidas siempre es menor al número de exitosas, y siendo una cantidad mucho menor con respecto a Round Robin, la segunda es que Weighted east-connection tiene fallos hasta el final, contrario del otro algoritmo que después del minuto 5 no registra ninguna transacción fallida.

Gráfica 7. Resumen del total de respuestas de las peticiones clasificadas en tiempos de respuesta (en milisegundos) usando el algoritmo Round Robin para 1000 usuarios.



Gráfica 8. Resumen del total de respuestas de las peticiones clasificadas en tiempos de respuesta (en milisegundos) usando el algoritmo Weighted east-connection para 1000 usuarios.



Las gráficas 7 y 8, muestra cada una una gráfica en barras en las cuales se puede evidenciar el número total de respuestas de las peticiones realizadas por los 1000 usuarios clasificadas en intervalos de tiempos de respuesta en milisegundos y adicional una barra para las peticiones que terminaron en error. A primera vista podría parecer información que ya hemos podido evidenciar en gráficas anteriores sin embargo una mirada más exhaustiva puede evidenciar que está gráfica dá una cantidad más precisa de este número de respuestas, ya que por ejemplo, aunque en la gráfica 3 y 4 se puede dar información similar respecto al tiempo de respuesta, hay que tener en cuenta que esa información está promediada ya que su objetivo se centra en ver el cambio de este tiempo de respuesta en torno a la duración de la prueba de 6 minutos, mientras que en estas dos gráficas de pastel esa información es omitida, ya que su objetivo se centra en mostrar las diferentes cantidades de peticiones en tiempos de respuesta discretizados. Gracias a esto podemos evidenciar nueva información la cual se expondrá a continuación.

En el algoritmo Round Robin podemos observar que no existen tiempos de respuesta en intervalos entre 500 y 1500 ms y los menores a 500 ms son casi nulos. Mientras que la mayoría de tiempos de respuestas son con tiempos superiores a los 1500 ms, siendo la barra que gana con gran diferencia la de respuestas fallidas, esto último ya lo pudimos observar en la gráfica 1 en donde se podía apreciar que para mil usuarios el porcentaje de fallos está por encima del 88%. Por otro lado al observar la gráfica 8, se puede evidenciar la gran mejora en cuestión de tiempos de respuesta que tiene el algoritmo Weighted east-connection, ya que podemos ver como la gran mayoría de las peticiones tienen un tiempo de respuesta por debajo de los 500 ms, el resto de las barras están bastante bajas siendo la más baja las cantidades por encima de los 1500 ms, aunque la de errores tiene una cantidad casi idéntica a la de tiempo de respuesta entre 500 y 1500 ms.

VII. CONCLUSIONES

- Teniendo en cuenta la implementación y configuración del proyecto podemos apreciar los diferentes algoritmos que se pueden utilizar para redireccionar el tráfico de las peticiones realizadas al servidor, una decisión acertada por parte del colectivo fue dividir los algoritmos en dos diferentes servidores para una mejor interpretación de los resultados, ya que se pueden hacer pruebas con el JMeter de manera individual, sin tener que hacerlas directamente por ambos algoritmos.

- Por otra parte se puede ver en gran medida la diferencia de cómo cada uno de los dos balanceadores resuelven las peticiones, ya que uno lo hace de manera equilibrada sin importar las características de los servidores y el otro las toma en cuenta dependiendo del tráfico que se esté realizando, enviandolas al servidor con menor cantidad de peticiones

- Hablando de la configuración del servidor, este se ha realizado haciendo uso de herramientas vistas anteriormente durante el desarrollo de las clases como los paquetes httpd, que en este vienen incluidos los módulos con los que trabajamos (mod_proxy_balancer), byrequest y bytraffic, que son los dos algoritmos empleados. Además de esto para mantener la configuración igual en las máquinas de cada integrante del grupo se utilizó Vagrant Cloud, en donde se montaron cada una de las máquinas y descargarlas y configurarlas mediante el archivo de configuración Vagrantfile, una herramienta muy práctica y de gran valor a la hora de trabajar en equipo.

- Con respecto al análisis de resultados obtenidos se puede evidenciar claramente que el mejor performance lo logra el algoritmo Weighted east-connection ya que no solo es claramente muy superior en tiempos de respuesta con respecto a Round Robin, sino también la gran cantidad de peticiones fallidas que genera Round Robin sobre todo al aumentar en gran medida el número de usuarios concurrentes lo hace un algoritmo no deseable para este tipo de necesidades.

- Hay dos cosas que se pueden tomar como positivas de Round Robin: lo primero es que su rendimiento es bastante estable, es decir no fluctúa tanto como Weighted east-connection, ya que en la mayoría de las pruebas se puede observar un gran pico que tuvo de disminución del performance a mitad de la duración de la prueba, por último, de las cosas positivas que se puede observar de Round Robin es que parece que tiende a mejorar su manejo de peticiones a medida de que el tiempo pasa, ya que se pudo observar una disminución de las peticiones con error que incluso llegó a cero después del minuto 4 y no volvió a aumentar.

REFERENCIAS

[1] NGINX, Inc. (2021, 26 enero). What Is Load Balancing? How Load Balancers Work. NGINX. Recuperado 6 de noviembre de 2021, de https://www.nginx.com/resources/glossary/load-balancing/

[2] The Apache Software Foundation. (s. f.). mod_proxy_balancer - Apache HTTP Server Version 2.4. Apache. Recuperado 6 de noviembre de 2021, de https://httpd.apache.org/docs/2.4/mod/mod_proxy_balancer.html

[3] Citrix, Inc. (s. f.). What is a Virtual Machine and How Does it Work? - Citrix Mexico. Citrix.com. Recuperado 6 de noviembre de 2021, de https://www.citrix.com/es-mx/solutions/vdi-and-daas/what-is-a-virtual-machine.html

[4] About the Editorial Staff. (s. f.). What is Apache? - What is a Web Server? WPBeginner. Recuperado 6 de noviembre de 2021, de https://www.wpbeginner.com/glossary/apache/

[5] GCFGlobal. (s. f.). Informática Básica: ¿Qué es un computador? GCFGlobal.org. Recuperado 6 de noviembre de 2021, de https://edu.gcfglobal.org/es/informatica-basica/que-es-un-computador/1/

[6] Fernández, Y. (2020, 1 junio). VirtualBox: qué es y cómo usarlo para crear una máquina virtual con Windows u otro sistema operativo. Xataka. Recuperado 6 de noviembre de 2021, de https://www.xataka.com/basics/virtualbox-que-como-usarlo-para-crear-maquina-virtual-windows-u-otro-sistema-operativo

[7] Ionos. (2020, 30 enero). ¿Qué es CentOS? Versiones CentOS y requisitos del sistema. IONOS Digitalguide. Recuperado 6 de noviembre de 2021, de https://www.ionos.es/digitalguide/servidores/know-how/que-es-centos-versiones-y-requisitos-del-sistema/

[8] An overview of HTTP - HTTP | MDN. (s. f.). MDN Web Docs. Recuperado 6 de noviembre de 2021, de https://developer.mozilla.org/en-US/docs/Web/HTTP/Overview

[9] What is a web server? - Learn web development | MDN. (s. f.). MDN Web Docs. Recuperado 6 de noviembre de 2021, de https://developer.mozilla.org/en-US/docs/Learn/Common_questions/What_is_a_web_server