

**Barranco Bailón, Manuel Alejandro
Garrido Mellado, Francisco Javier
Servidores Web de Altas Prestaciones**

Balanceadores de carga y algoritmos de balanceo

Índice

1.- Introducción

2.- Balanceadores de carga software

3.- Estudio y uso de diferentes balanceadores

3.1.- NGINX

3.2.- ZenLoad

3.3.- Octopus

4.- Conclusiones

4.1.- Medidas con siege

4.2.- Medidas con ab

4.3.- Ejemplos de las pruebas

5.- Bibliografía

1.- Introducción

En el presente documento se recogen diferentes balanceadores de carga software de código abierto con mayor relevancia en la actualidad. Además, se especifican con mayor detalle tres de ellos, en los cuales se explica detalladamente su instalación, configuración y uso. Por último, en ellos se realizarán mediciones para visualizar los distintos resultados obtenidos mediante diferentes algoritmos de balanceo.

2.- Balanceadores de carga software

Un balanceador de carga es un dispositivo hardware o software que se pone al frente de un conjunto de servidores que atienden una aplicación y, como su nombre indica, asigna o balancea las solicitudes que llegan de los clientes a los servidores finales usando algún algoritmo(Round Robin, pesos, prioridad, menor conexiones, menor carga, etc) para evitar los así denominados cuellos de botella.

Entre los diferentes algoritmos se encuentran:

- Round Robin : algoritmo de balanceo basado en una cola de turnos. Adecuado si todos tienen potencia similar.
- Menor número de conexiones (Least Connections): se reparte el trabajo según la utilización de cada una (el balanceador lleva la cuenta de cada uno). Adecuado para distribuir el trabajo entre máquinas similares, evitándose la sobrecarga de las que puedan tener mas trabajo.
- Ponderación (Weight) : se reparte trabajo según un peso asignado a cada máquina. Adecuado para cuando los dos anteriores algoritmos no den buen resultado, es necesario ajustarlo según las capacidades de las máquinas.
- Prioridad : similar a la ponderación pero con grupos de máquinas. Cada grupo tiene una prioridad y un máximo de conexiones simultaneas que puede atender.
- Tiempo de respuesta : se reparte el trabajo según los tiempos de respuesta de cada máquina. El balanceador debe de estar monitorizando las máquinas.
- Combinación : se reparte el trabajo usando una mezcla de los algoritmos anteriores. Es el mas eficientes. Por ejemplo, ZenLoad usa un algoritmo basado en tiempo de respuesta y menor número de conexiones.

Los balanceadores pueden realizar tareas adicionales como comprobar la disponibilidad de servidores, proteger de diversos ataques y derivar en función del tráfico.

Entre los balanceadores hardware y software se diferencian en que los primeros son más eficientes aunque menos configurables y más costosos. En cambio los balanceadores software permiten una mayor flexibilidad y existe un gran número de ellos gratuitos.

A continuación se nombran algunos de los balanceadores software más conocidos en la actualidad:

- **Linux Virtual Server** → <http://www.LinuxVirtualServer.org/>

Implementa los siguientes algoritmos:

- Planificación por Round-Robin (Round-Robin Scheduling).
- Planificación por Round-Robin ponderado (Weighted Round-Robin Scheduling).
- Planificación por menor número de conexiones (Least-Connection Scheduling).
- Planificación por menor número de conexiones ponderado (Weighted Least-Connection Scheduling).
- Planificación por menor número de conexiones local (Locality-Based Least-Connection Scheduling).
- Planificación por menor número de conexiones local con réplicas (Locality-Based Least-Connection with Replication Scheduling).
- Planificación por hashing de destino (Destination Hashing Scheduling).
- Planificación por hashing de origen (Source Hashing Scheduling).
- Planificación por menor retardo esperado (Shortest Expected Delay Scheduling).
- Planificación por servidores sin peticiones en espera (Never Queue Scheduling).

- **BalanceNG** → <http://www.inlab.de/balanceng/>

Implementa los siguientes algoritmos:

- Round Robin
- Simple Weighted Round Robin Random
- Weighted Random
- Client Address Hashing
- Least Session
- Least Bandwidth
- Least Resource based on agent supplied information.

Haproxy → <http://haproxy.1wt.eu/>

Implementa los siguientes algoritmos:

- Round Robin
- Static-rr
- Leastconn
- First
- Source
- Uri
- url_param
- hdr(<name>)
- rdp_cookie

Pen → <http://siag.nu/pen/>

Implementa los siguientes algoritmos:

- Round Robin
- Weight
- Last time
- Max number of clients

Crossroads Load Balancer → <http://crossroads.e-tunity.com/>

Implementa los siguientes algoritmos:

- Round Robin
- Least connections
- First available
- External Program
- Strict-Hashed-ip
- Lax-Hashed-ip
- Strict-Stored-ip (last time)
- Lax-Stored-ip
- Weight

Distributor load balancer → <http://distributor.sourceforge.net/>

Implementa los siguientes algoritmos:

- Round Robin
- Hash
- Httpcookie

No se describen los distintos balanceadores porque prácticamente tienen la misma funcionalidad exceptuando pequeños detalles que se pueden consultar en sus respectivos enlaces.

3.- Estudio y uso de diferentes balanceadores

En este apartado se muestran las instalaciones, configuraciones y los distintos resultados obtenidos en algunos balanceadores con determinadas herramientas de medición de rendimiento.

3.1.- Escenario

Se ha planteado un escenario con una máquina balanceadora y cuatro servidores finales, de manera que la máquina balanceadora ha usado Nginx, ZenLoad u Octopus para cada prueba respectivamente.

Los servidores finales corren bajo un Ubuntu 12.04 Server con LAMP instalado y sirviendo una página web dinámica que consta de un bucle de cincuenta mil vueltas.

Una tabla que resume los servidores usados con su correspondiente IP y memoria RAM es la siguiente:

	Nginx	ZenLoad	Octopus	Nodo-1	Nodo-2	Nodo-3	Nodo-4
IP	192.168.2.130	192.168.2.134	192.168.2.136	192.168.2.128	192.168.2.129	192.168.2.132	192.168.2.133
RAM	512MB	512Mb	512MB	512MB	512MB	256MB	128MB
Number of Processor core	1	1	1	1	1	1	1

3.2.- NGINX (<http://wiki.nginx.org/NginxEs>)

3.2.1.- Instalación

- Importar la clave del repositorio.
 - `cd /tmp/`
 - `wget http://nginx.org/keys/nginx_signing.key`
 - `apt-key add /tmp/nginx_signing.key`
 - `rm -f /tmp/nginx_signing.key`
- Añadir el repositorio al fichero `/etc/apt/sources.list`
 - `echo "deb http://nginx.org/packages/ubuntu/ lucid nginx" >> /etc/apt/sources.list`
 - `echo "deb-src http://nginx.org/packages/ubuntu/ lucid nginx" >> /etc/apt/sources.list`
- Instalar el paquete de nginx
 - `apt-get update`
 - `apt-get install nginx`

3.2.2.- Configuración

- Editar el fichero `/etc/nginx/conf.d/default.conf`.
- Configurar los servidores finales que forman la granja web. En el ejemplo que se muestra a continuación, se nombra al grupo de servidores como “apaches”, pudiendo llevar cualquier otro nombre.

```
upstream apaches {
    server 172.16.168.130;
    server 172.16.168.131;
}
```

Si no se indica nada específico en la directiva `upstream` como ocurre en este ejemplo, el algoritmo de balanceo usado por defecto será Round Robin.

- Indicar a nginx que use el grupo de servidores definido en el punto anterior, así como el puerto de escucha y el nombre del servidor que actúa como balanceador. En la directiva `location`, indicamos que la conexión entre el balanceador nginx y los servidores finales se use HTTP 1.1 y se eliminen las cabeceras `Connection` para evitar que se pase al servidor final la cabecera del cliente.

```
server{
    listen 80;
    server_name balanceador;
    access_log /var/log/nginx/balanceador.access.log;
    error_log /var/log/nginx/balanceador.error.log;
    root /var/www/;
    location /
    {
        proxy_pass http://apaches;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_http_version 1.1;
    }
}
```

```

        proxy_set_header Connection "";
    }
}

```

A continuación, se mostrará la configuración para los algoritmos de peso y de menor número de conexiones, que como se dijo anteriormente, se especificará en la directiva upstream.

3.2.2.1.- Peso

A cada servidor del grupo de servidores establecido en la directiva upstream, se le asignará un peso específico mediante el modificador weight.

```

##### definicion de mis dos maquinas apache

    upstream apaches{
        server 192.168.2.128 weight=4;
        server 192.168.2.129 weight=4;
        server 192.168.2.132 weight=2;
        server 192.168.2.133 weight=1;
        keepalive 3;
    }

#####

```

En la configuración que establecimos para nuestro escenario, asignamos esos pesos a cada uno de los servidores porque los dos primeros (.128 y .129 con 512 MB de RAM cada una) tienen cuatro veces la memoria RAM que la última máquina (.133 con 128 MB de RAM) y el doble que la tercera (.132 con 256 MB de RAM). De esta forma, de cada 11 peticiones que lleguen al balanceador, 4 irán para la máquina .128, otras 4 por la .129, 2 por la .132 y 1 por la .133 .

3.2.2.2.- Menor número de conexiones

El balanceador reparte las solicitudes a los servidores finales según la que tenga menor número de conexiones establecidas sirviendo contenido. Para este algoritmo, bastará con añadir el parámetro least_conn;

```

##### definicion de mis dos maquinas apache

    upstream apaches{
        least_conn;
        server 192.168.2.128;
        server 192.168.2.129;
        server 192.168.2.132;
        server 192.168.2.133;
        keepalive 3;
    }

#####

```

3.3.- ZenLoad (www.zenloadbalancer.com)

Uno de los balanceadores software más relevantes en la comunidad por sus prestaciones, facilidad de configuración por su interfaz gráfica sencilla y presentar diversas versiones gratuitas y de pago. Está basado en la configuración Debian y ofrece los algoritmos de balanceo de Round Robin, hash, pesos, prioridad.

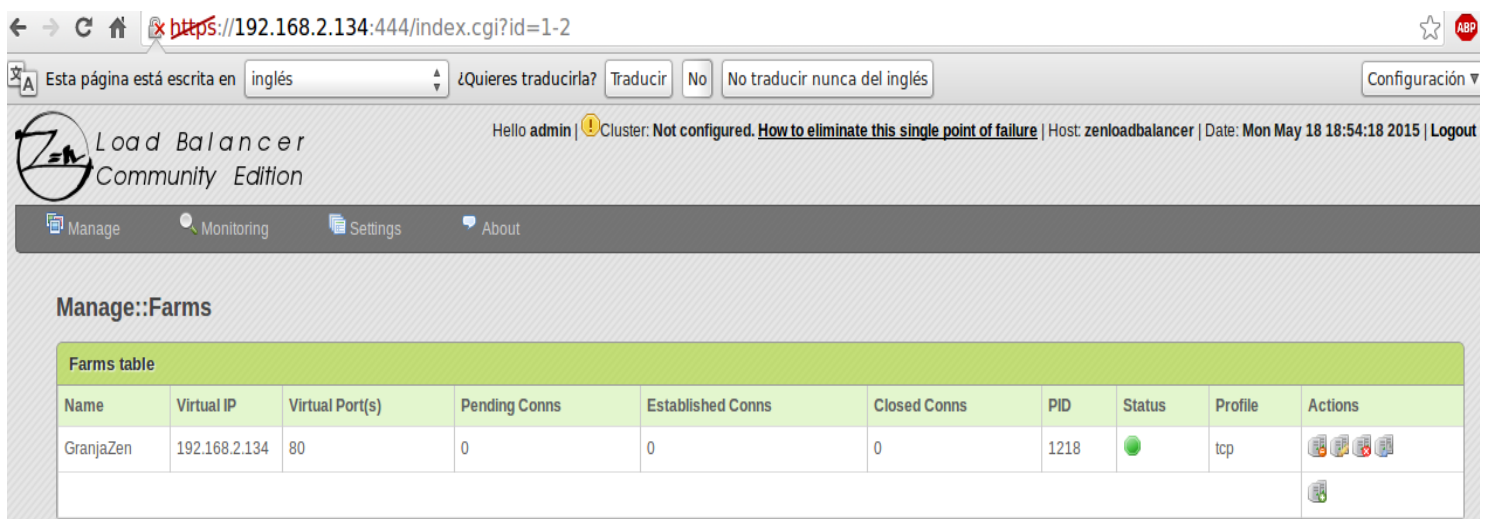
Señalar que en el algoritmo por pesos, Zen Load calcula primeramente en base a la carga actual de trabajo de cada servidor, y sobre esta medida aplica de manera lineal según la asignación de pesos.

3.3.1.- Instalación

- Puede instalarse con una imagen ISO de Debian con Zen ya instalado, o con un paquete tar.gz. En nuestro caso, se optó por la primera opción siguiendo los pasos de cualquier distribución Linux.

3.3.2.- Configuración














- Se accede a través de un navegador web, a través de la dirección IP de la máquina balanceadora y el puerto 444, siendo user y password, por defecto, admin.
- En la pestaña Manage → Farms, se establecerá la máquina balanceadora así como los demás servidores finales que formarán la granja web.









The screenshot shows the ZenLoadbalancer web interface. The browser address bar displays <https://192.168.2.134:444/index.cgi?id=1-2>. The page header includes the ZenLoadbalancer logo, the text "Load Balancer Community Edition", and a status message: "Hello admin | Cluster: Not configured. [How to eliminate this single point of failure](#) | Host: zenloadbalancer | Date: Mon May 18 18:54:18 2015 | Logout". The navigation bar contains links for "Manage", "Monitoring", "Settings", and "About". The main content area is titled "Manage::Farms" and displays a table with the following data:

Name	Virtual IP	Virtual Port(s)	Pending Conns	Established Conns	Closed Conns	PID	Status	Profile	Actions
GranjaZen	192.168.2.134	80	0	0	0	1218		tcp	

Pulsando sobre editar, se procede a definir los servidores finales que forman la granja como se observa en la siguiente imagen:

Edit real IP servers configuration						
Server	Address	Port	Max connections	Weight	Priority	Actions
0	192.168.2.128	80	1000	4	0	  
1	192.168.2.129	80	1000	4	0	  
2	192.168.2.132	80	1000	2	0	  
3	192.168.2.133	80	1000	1	0	  
						

Cancel

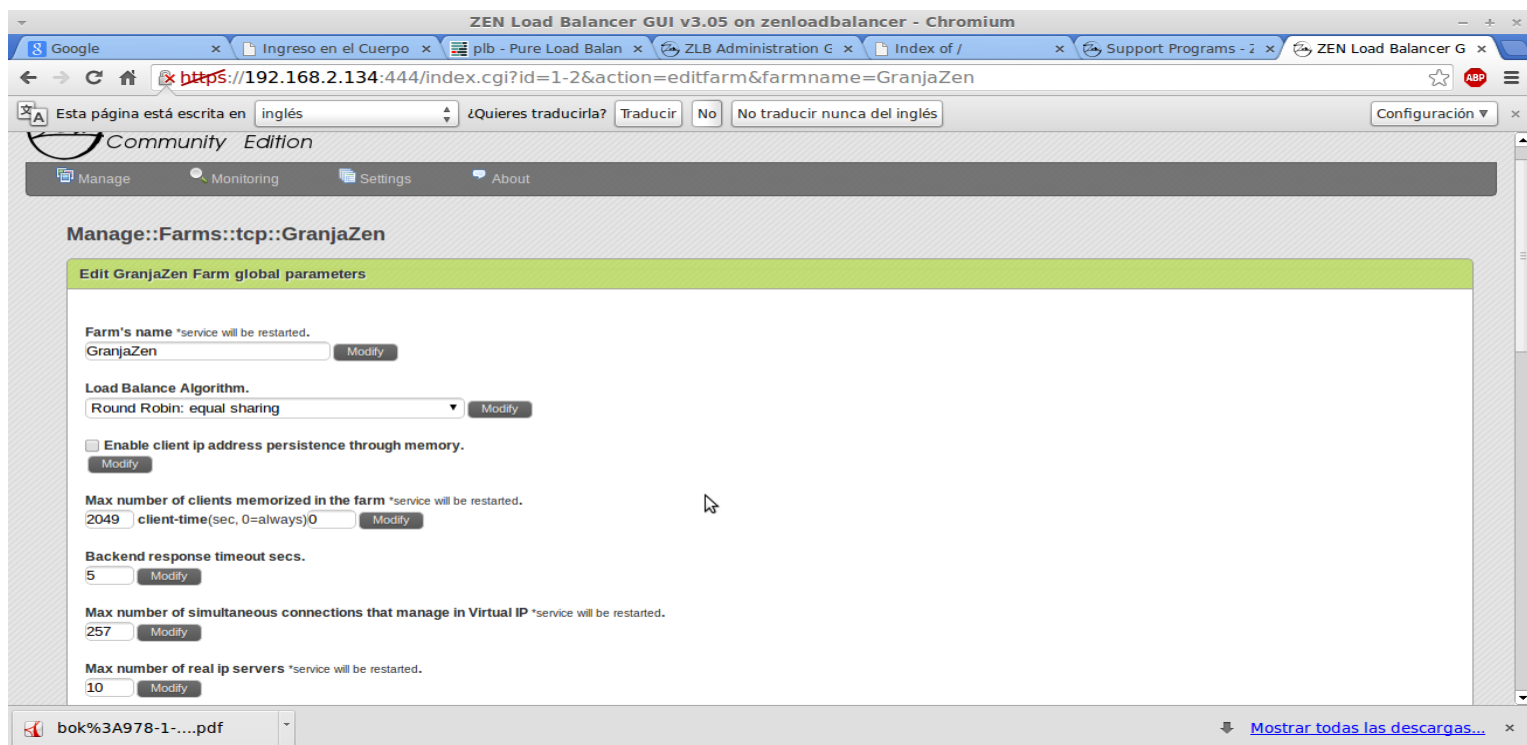
Farms table									
Name	Virtual IP	Virtual Port(s)	Pending Conns	Established Conns	Closed Conns	PID	Status	Profile	Actions
GranjaZen	192.168.2.134	80	0	0	0	1218		tcp	   
									

Al definir las máquinas no será necesario indicarle pesos a cada una de ellas a no ser que se haya indicado dicho algoritmo.

Para especificar el algoritmo de balanceo, también se realizará en la opción Manage → Farms → icono Editar.

3.3.2.1.- Round Robin

Al especificar que el algoritmo a usar será Round Robin, el parámetro Weight indicado en el apartado anterior, se establece automáticamente al valor 1 para todos los servidores y así realizar el reparto por turnos.



ZEN Load Balancer GUI v3.05 on zenloadbalancer - Chromium

https://192.168.2.134:444/index.cgi?id=1-2&action=editfarm&farmname=GranjaZen

Manage Monitoring Settings About

Manage::Farms::tcp::GranjaZen

Edit GranjaZen Farm global parameters

Farm's name *service will be restarted.

Load Balance Algorithm.

☐ Enable client ip address persistence through memory.

Max number of clients memorized in the farm *service will be restarted.
 client-time(sec, 0=always)

Backend response timeout secs.

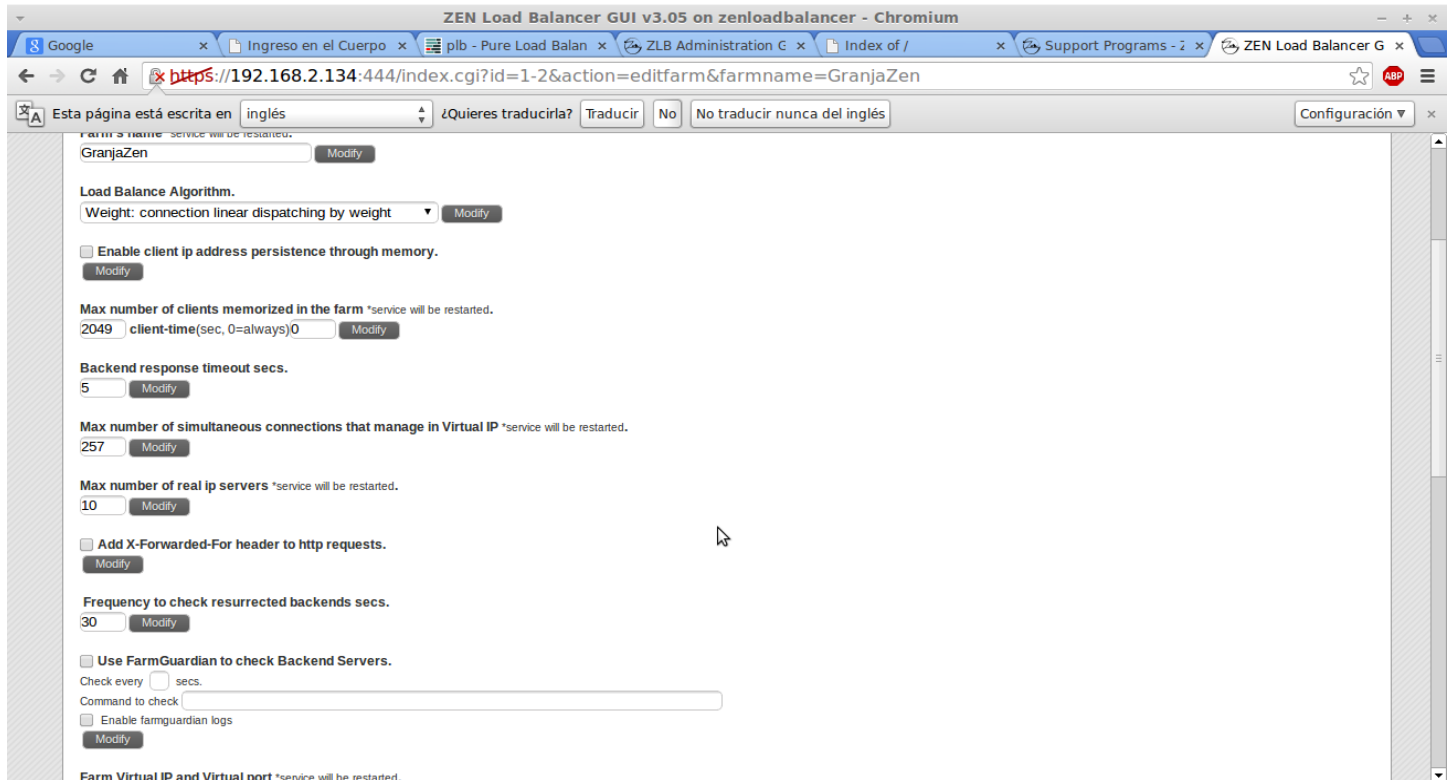
Max number of simultaneous connections that manage in Virtual IP *service will be restarted.

Max number of real ip servers *service will be restarted.

bok%3A978-1-....pdf [Mostrar todas las descargas...](#)

3.3.2.2.- Pesos

Teniendo en cuenta la manera en la que Zen interpreta este algoritmo comentada anteriormente (carga + peso) , y que el reparto de pesos la hemos distribuido de igual forma que se explicó en el balanceador nginx por la misma causa de las proporciones de memoria RAM de cada una de los servidores, la configuración quedaría de la siguiente forma:



3.4.- Octopus Load Balancer (<http://sourceforge.net/projects/octopuslb/>)

El otro balanceador que se probó en nuestro estudio es Octopus Load Balancer, que se trata de un balanceador software de código abierto para Linux.

Octopus nos permite establecer los siguientes algoritmos de balanceo:

- Least Load (LL)
- Least Connections (LC)
- Round Robin (RR)
- http URI Hashing (HASH)
- static URI Hashing (STATIC)

En nuestro caso se ha optado por las opciones de Round Robin y Least Connections, la opción Least Load requería de la instalación de la librería SNMP.

3.4.1.- Instalación

- La instalación se realiza mediante un tar.gz , para ello se descomprime y se procede a leer el archivo INSTALL donde se detalla cada paso de la instalación. Estos son los siguientes:
 - \$./configure --disable-snmp (if compiling without SNMP support)
 - \$ make
 - \$ make install

Como se indica, si la librería SNMP no está instalada se optará por estos pasos, en caso contrario (habilitaría el algoritmo Least Load) pueden seguirse estos:

- tar zxvf octopuslb-1.14.tar.gz
- \$ cd octopuslb-1.14s
- \$./configure (if you're compiling with SNMP support)

3.4.2.- Configuración

Para la configuración de los diferentes parámetros del balanceador se accede al archivo de configuración /usr/local/etc/octopuslb.conf donde se ha establecido los siguientes parámetros para su funcionamiento:

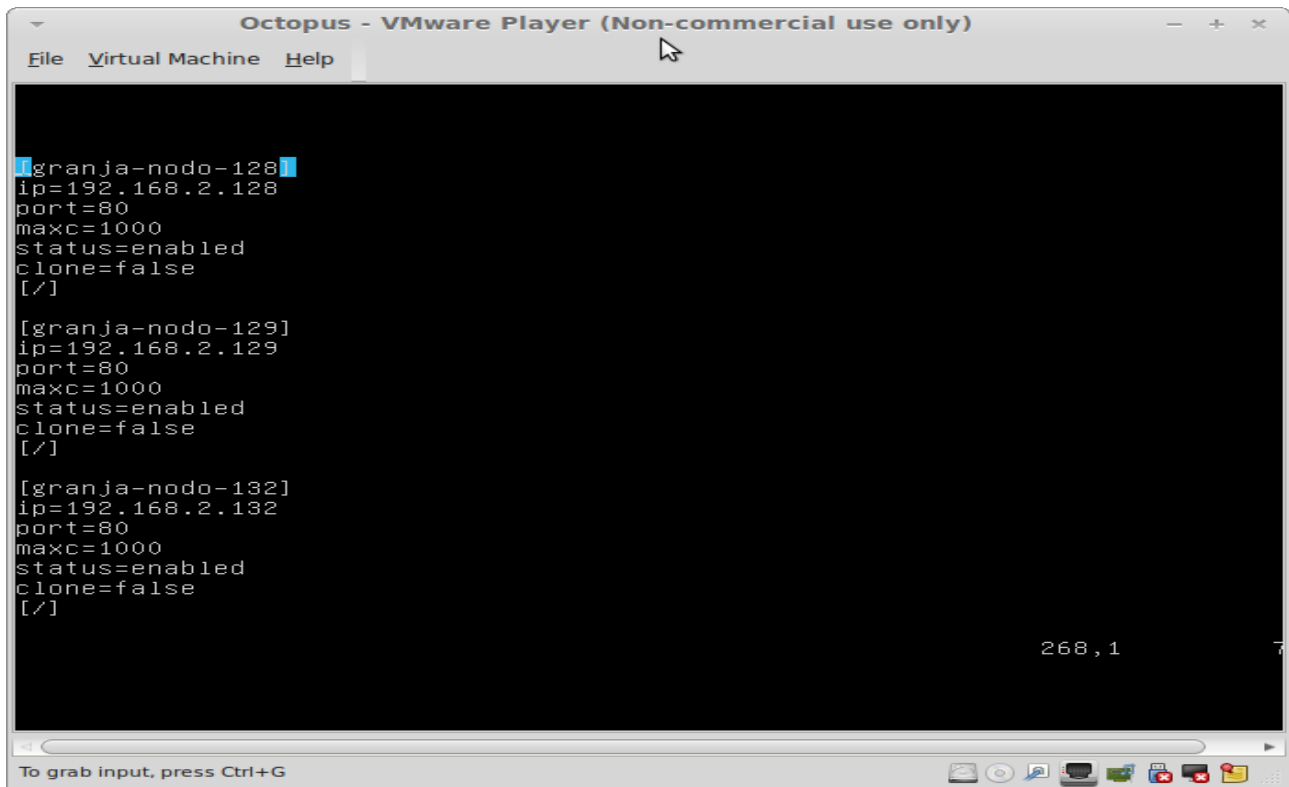
```
binding_port=80
algorithm=RR
default_maxc=1000
shm_run_dir=/var/run/octopuslb
log_file=/var/log/octopuslb.log
```

```
[granja-nodo-128]
ip=192.168.2.128
port=80
maxc=1000
status=enabled
clone=false
[/]
```

```
[granja-nodo-129]
ip=192.168.2.129
port=80
maxc=1000
status=enabled
clone=false
[/]
```

```
[granja-nodo-132]
ip=192.168.2.132
port=80
maxc=1000
status=enabled
clone=false
[/]
```

```
[granja-nodo-133]
ip=192.168.2.133
port=80
maxc=1000
status=enabled
clone=false
[/]
```



```
[granja-nodo-128]
ip=192.168.2.128
port=80
maxc=1000
status=enabled
clone=false
[/]

[granja-nodo-129]
ip=192.168.2.129
port=80
maxc=1000
status=enabled
clone=false
[/]

[granja-nodo-132]
ip=192.168.2.132
port=80
maxc=1000
status=enabled
clone=false
[/]

268,1
```

El parámetro *binding_port* es el puerto de escucha del balanceador, *algorithm* es el parámetro que recoge el algoritmo que emplea el balanceador, se dispone de RR (Round Robin), LC (Least Connections), LL (Least Load), HASH HTTP URI y STATIC (muy similar al anterior), el parámetro *default_maxc* define el máximo de número de sesiones del servidor. Por último, lo más importante, se definen los servidores finales que forman la granja web siguiendo la siguiente esquemática:

```
[granja-nodo-132]
ip=192.168.2.132
port=80
maxc=1000
status=enabled
clone=false
[/]
```

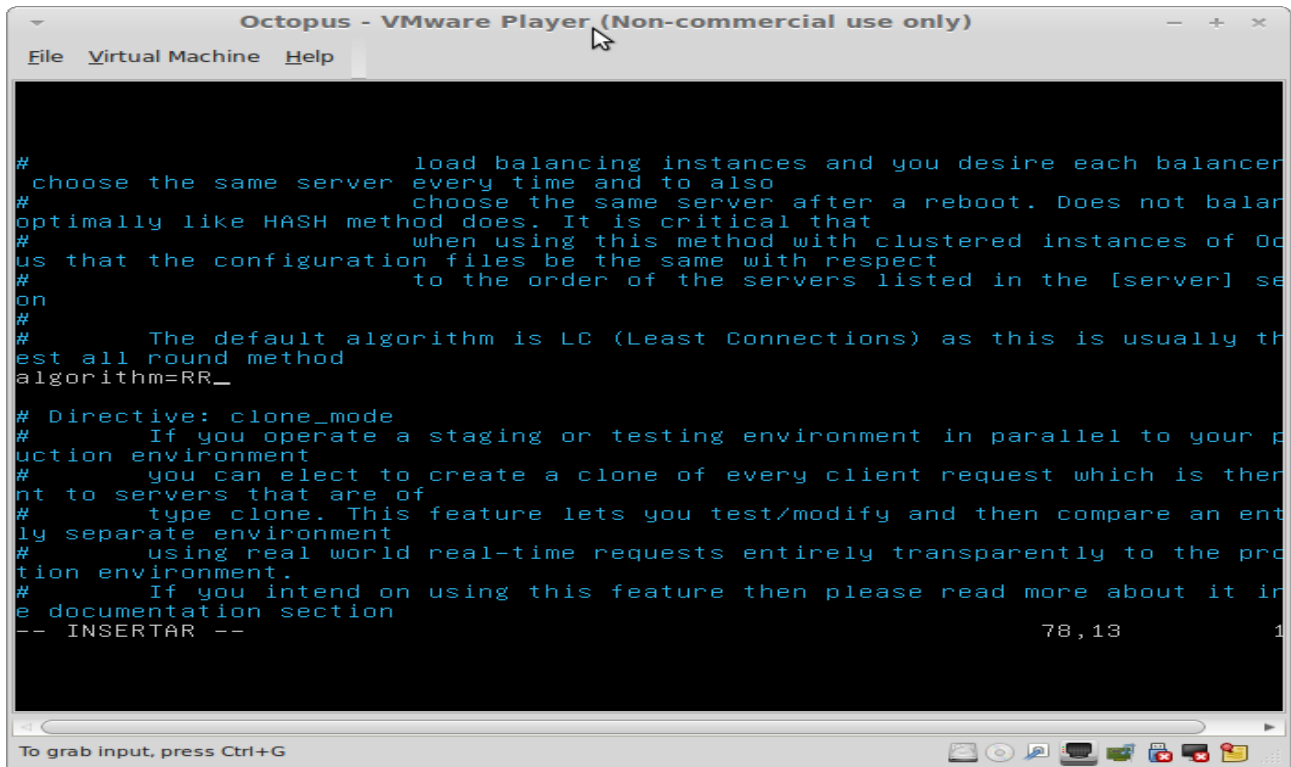
Entre corchetes el nombre del servidor, a continuación la directiva *ip* donde se define su correspondiente dirección, *port* es el puerto de escucha de dicho servidor, *maxc* el número máximo de sesiones, *status* indica que el servidor esta habilitado y en *clone* si hace de máquina de apoyo por si algún servidor cae.

Existen muchas mas opciones pero se han dejado por defecto (no es necesario descomentar nada).

Por último para arrancar el servicio se ejecuta **octopuslb-server -c /usr/local/etc/octopuslb.conf**

3.3.2.1.- Round Robin

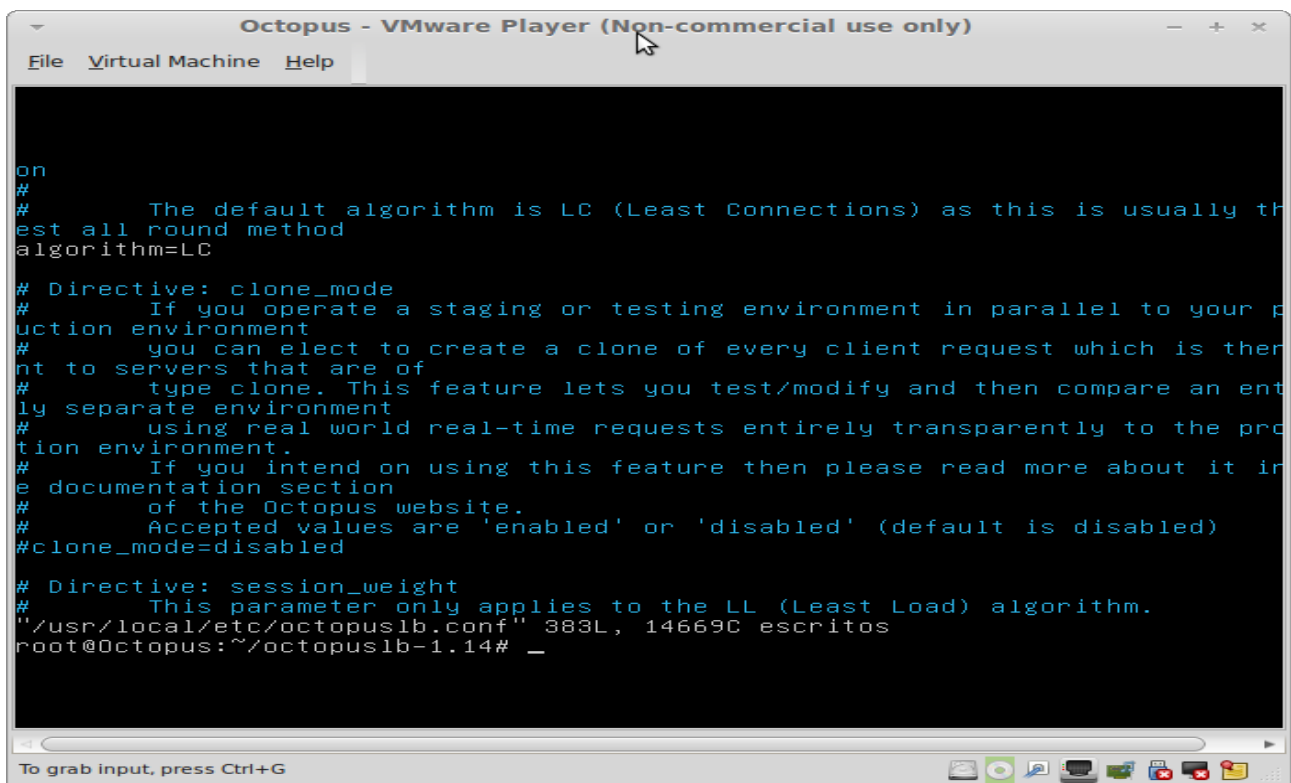
Como se indicó anteriormente solo es necesario cambiar un parámetro.



The screenshot shows a terminal window titled "Octopus - VMware Player (Non-commercial use only)". The terminal displays a configuration file for Octopus. The relevant lines are:

```
# choose the same server every time and to also
# choose the same server after a reboot. Does not balance
# optimally like HASH method does. It is critical that
# when using this method with clustered instances of Octopus
# that the configuration files be the same with respect
# to the order of the servers listed in the [server] section
on
#
# The default algorithm is LC (Least Connections) as this is usually the
# best all round method
algorithm=RR_
```

3.3.2.2.- Menor número de conexiones



The screenshot shows a terminal window titled "Octopus - VMware Player (Non-commercial use only)". The terminal displays a configuration file for Octopus. The relevant lines are:

```
on
#
# The default algorithm is LC (Least Connections) as this is usually the
# best all round method
algorithm=LC

# Directive: clone_mode
# If you operate a staging or testing environment in parallel to your production
# environment you can elect to create a clone of every client request which is then
# sent to servers that are of type clone. This feature lets you test/modify and then compare an entirely
# separate environment using real world real-time requests entirely transparently to the production
# environment. If you intend on using this feature then please read more about it in the
# documentation section of the Octopus website.
# Accepted values are 'enabled' or 'disabled' (default is disabled)
#clone_mode=disabled

# Directive: session_weight
# This parameter only applies to the LL (Least Load) algorithm.
"/usr/local/etc/octopuslb.conf" 383L, 14669C escritos
root@Octopus:~/octopuslb-1.14# _
```

Solo es necesario ponerle el parámetro LC.

3.5 Diferencias entre Nginx, ZenLoad y Octopus

Como se aprecia hay diferencias sutiles entre unos y otros siendo la finalidad la misma, entre las diferencias existentes hay tres muy importante que son la instalación, su configuración y los servicios ofrecidos.

Referente a la instalación, la facilidad que ofrece ZenLoad es difícilmente alcanzable por otro balanceador.

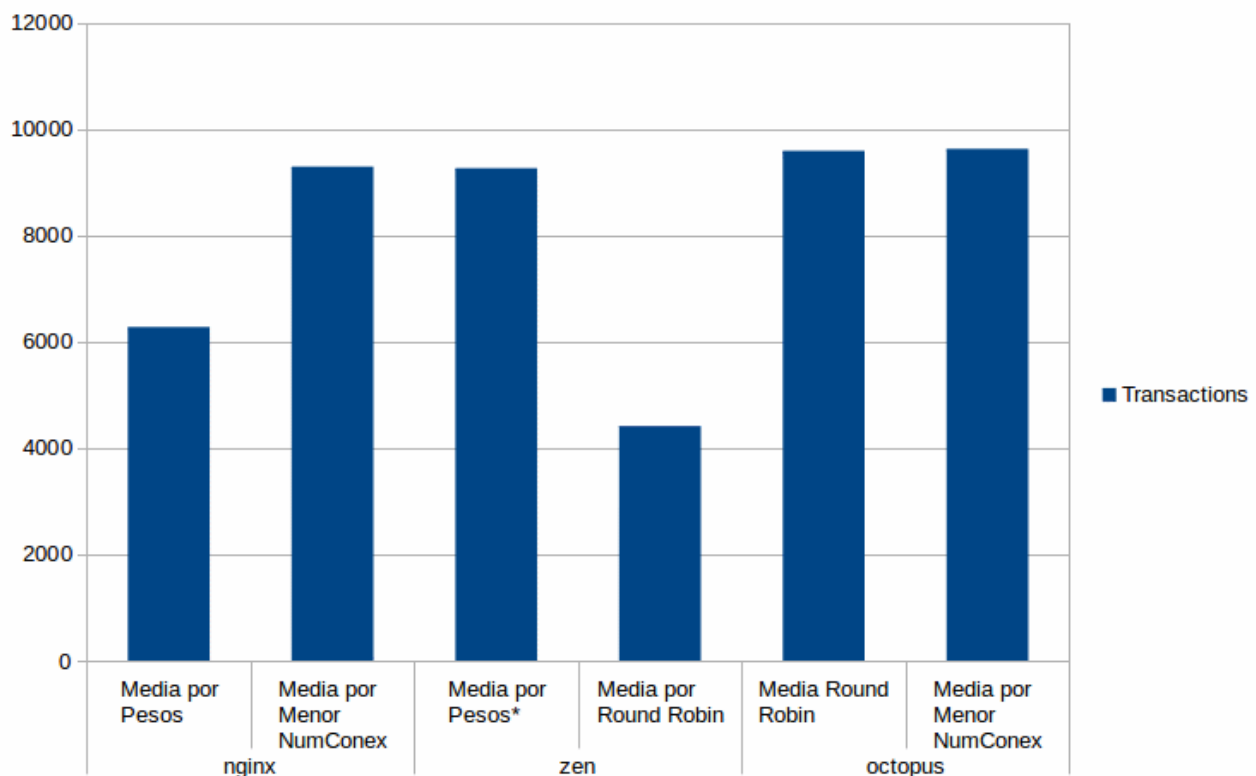
En cuanto a la configuración nuevamente ZenLoad gana la batalla, siendo tremendamente fácil la configuración de esta.

Por ultimo, los tres no ofrecen los mismos servicios, con Nginx se puede probar los algoritmos Round Robin, Least connections (menor número de conexiones), hash y por pesos , mientras que ZenLoad ofrece Round Robin , por pesos ,prioridad y hash, siendo el algoritmo de pesos como se comentó anteriormente diferente y no ofreciendo el algoritmo de menor número de conexiones a diferencia de Nginx y Octopus. Este último ofrece los algoritmos Round Robin, Least Connections, Least Load, Hash http uri y static, por tanto, según hemos observado Nginx y Octopus son mas flexibles a la hora de configurar que ZenLoad que no da mucho margen para ello.

4.Conclusiones

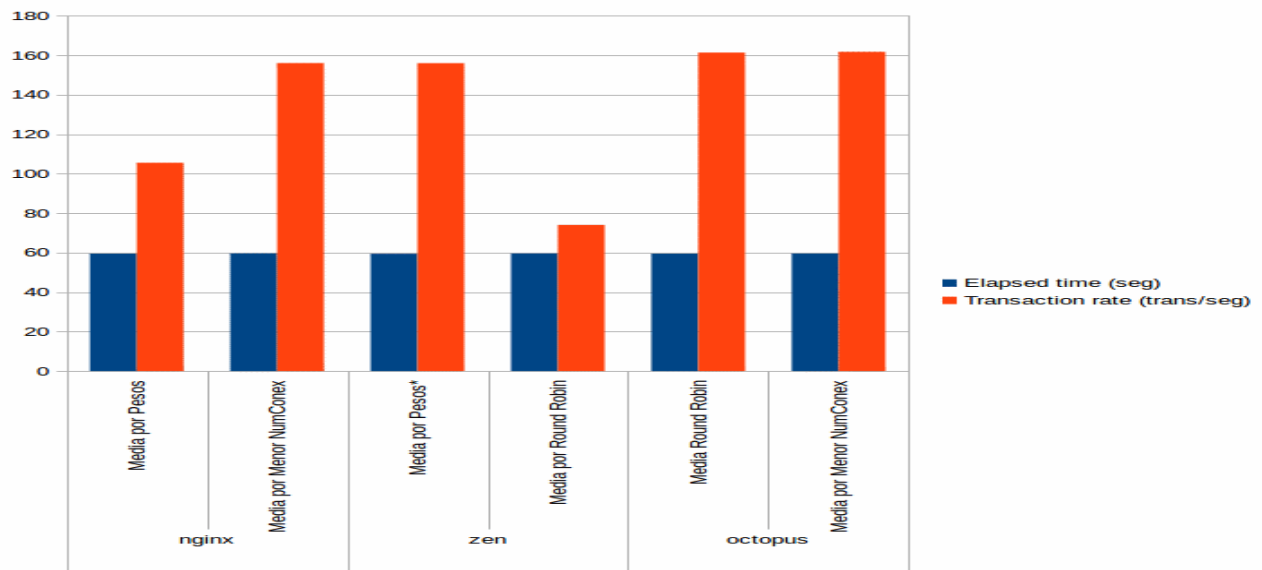
4.1 Medidas con Siege

-Transactions



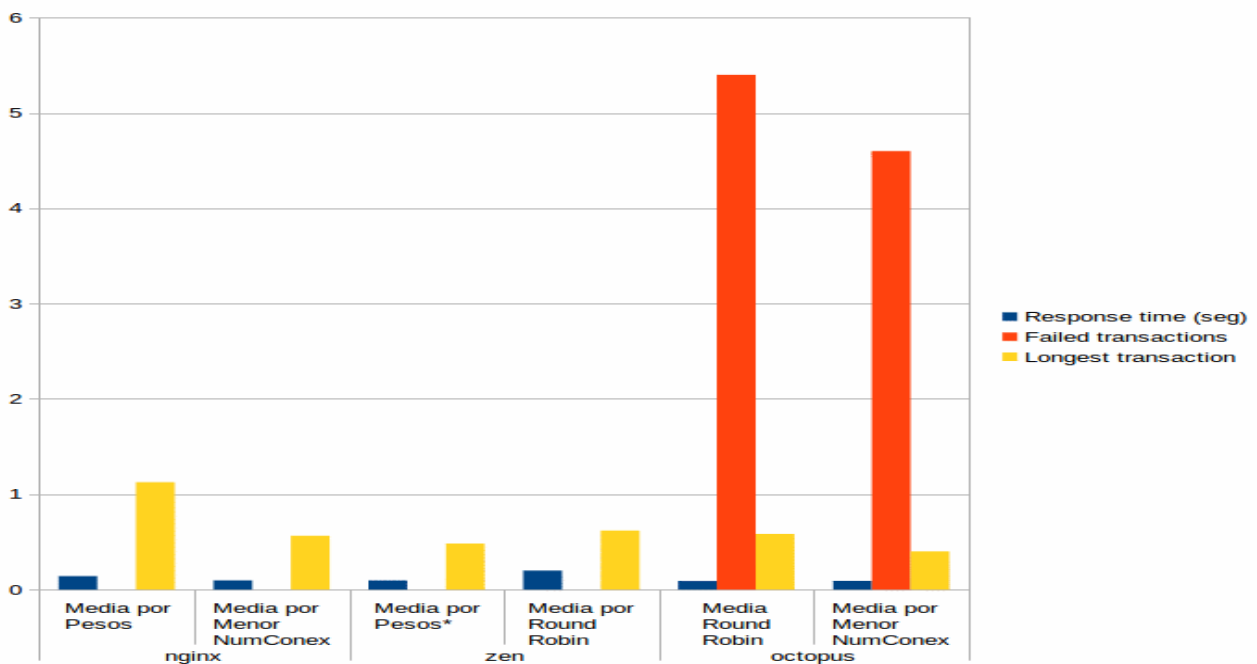
- **Menor numero de conexiones:** se observa como nginx y octopus tienen un resultado similar, posiblemente porque los algoritmos sean practicamente iguales.
- **Pesos:** En este caso ZenLoad ofrece un resultado mucho mejor lo cual es debido a que también implementa el algoritmo de menor carga.
- **Round Robin :** el resultado de ZenLoad en este caso es pobre, ofreciendo Octopus un muy buen rendimiento.

-Elapsed Time and Transaction Rate



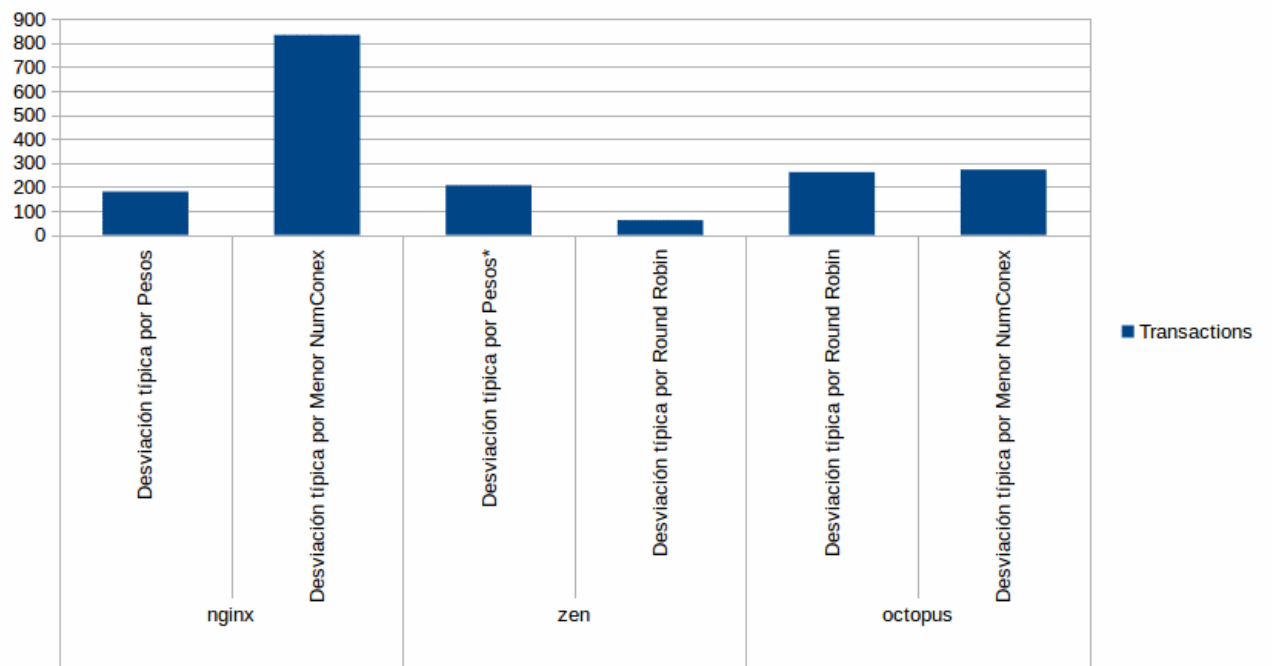
En este caso los resultados son muy similares salvo en el caso de media por Round Robin con ZenLoad y media por Pesos con Nginx donde empeoran.

-Response Time, Failed Transactions and Longest Transactions

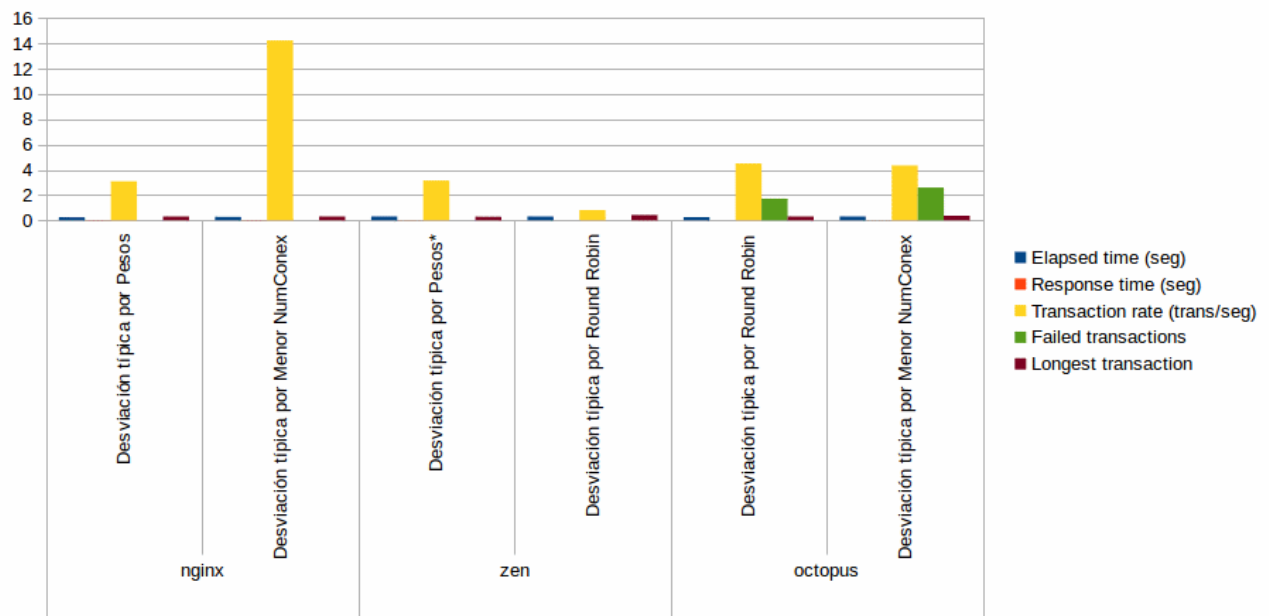


En este caso se arroja resultados similares salvo en Failed transactions donde hay un ligero incremento sin importancia con Octopus.

-Desviación Transactions

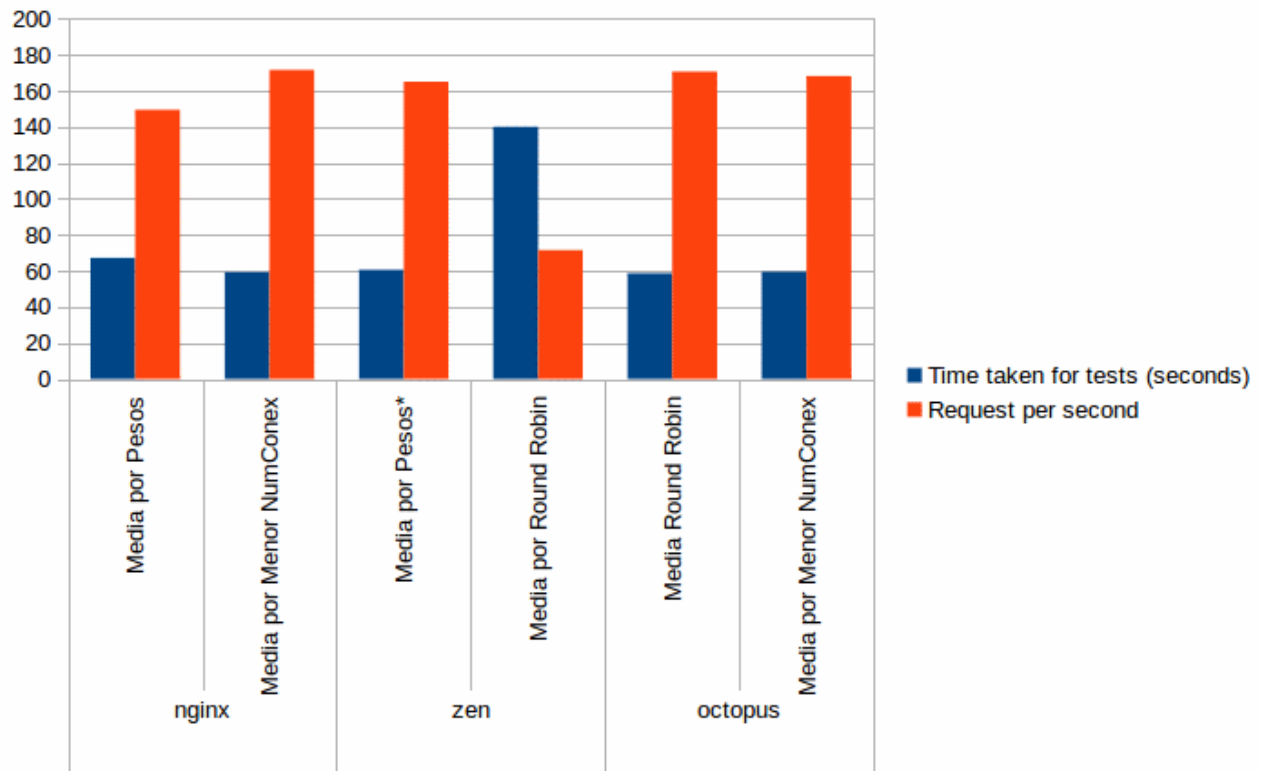


-Desviación Elapsed Time, Transaction Rate, Response Time, Failed Transactions and Longest Transactions



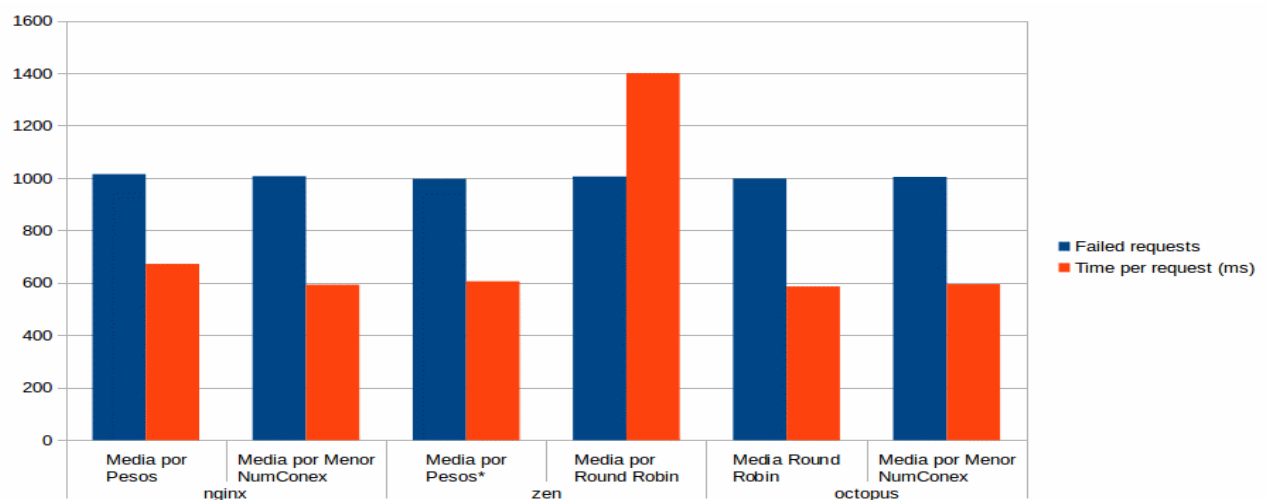
4.2 Medidas con ab

-Time taken for tests y Request per second



- **Menor numero de conexiones:** se observa como octopus y nginx tienen un resultado similar tanto para Request per second como para Time taken for tests.
- **Pesos:** En este caso ZenLoad ofrece un resultado ligeramente mejor para el campo Request per Second.
- **Round Robin :** el resultado de ZenLoad en este caso es pobre, ofreciendo Octopus un muy buen rendimiento.

-Failed requests and Time per request



- **Menor numero de conexiones:** se observa como octopus y nginx tienen un resultado similar.
- **Pesos:** En este caso ambos tienen un resultado similar.
- **Round Robin :** el resultado de ZenLoad en este caso es pobre, ofreciendo Octopus un mejor resultado.

4.3 Ejemplo de las pruebas

4.3.1 Ab con menor número de conexiones Nginx

```

Terminal
Archivo  Editar  Ver  Buscar  Terminal  Ayuda
Server Software:      nginx/1.6.2
Server Hostname:      192.168.2.130
Server Port:          80

Document Path:        /f.php
Document Length:      23 bytes

Concurrency Level:    100
Time taken for tests:  55.101 seconds
Complete requests:    10000
Failed requests:      1021
    (Connect: 0, Receive: 0, Length: 1021, Exceptions: 0)
Write errors:         0
Total transferred:    2248868 bytes
HTML transferred:     228868 bytes
Requests per second:  181.49 [#/sec] (mean)
Time per request:     551.009 [ms] (mean)
Time per request:     5.510 [ms] (mean, across all concurrent requests)
Transfer rate:        39.86 [Kbytes/sec] received

Connection Times (ms)
      min    mean[+/-sd] median   max
Connect:    0      1  0.8      0     10
Processing: 156    549 246.3    528   4647
Waiting:    155    549 246.3    528   4646
Total:      156    550 246.6    529   4656
WARNING: The median and mean for the initial connection time are not within a normal deviation
These results are probably not that reliable.

Percentage of the requests served within a certain time (ms)
 50%    529
 66%    567
 75%    596
 80%    616
 90%    674
 95%    731
 98%    811
 99%    872
100%   4656 (longest request)
javi@javi: ~/Escritorio/ESTUDIOS/3ºGrado Ingenieria Informatica/SEGUNDO CUATRIMESTRE/SERVIDORES_WEB/TRABAJO $

```

4.3.2 Ab con menor número de conexiones Octopus

```

Terminal
Archivo  Editar  Ver  Buscar  Terminal  Ayuda
Server Port:          80

Document Path:        /f.php
Document Length:      23 bytes

Concurrency Level:    100
Time taken for tests:  60.389 seconds
Complete requests:    10000
Failed requests:      997
    (Connect: 0, Receive: 0, Length: 997, Exceptions: 0)
Write errors:         0
Total transferred:    2358897 bytes
HTML transferred:     228897 bytes
Requests per second:  165.59 [#/sec] (mean)
Time per request:     603.895 [ms] (mean)
Time per request:     6.039 [ms] (mean, across all concurrent requests)
Transfer rate:        38.15 [Kbytes/sec] received

Connection Times (ms)
      min    mean[+/-sd] median   max
Connect:    0      0  0.8      0     12
Processing: 181    602 135.2    575   1555
Waiting:    180    573 133.6    547   1555
Total:      184    603 135.2    576   1556

Percentage of the requests served within a certain time (ms)
 50%    576
 66%    629
 75%    672
 80%    700
 90%    782
 95%    861
 98%    950
 99%   1024
100%   1556 (longest request)
javi@javi ~ $

```

4.3.3 Ab con RoundRobin ZenLoad

```
Terminal
Archivo  Editor  Ver  Buscar  Terminal  Ayuda

Server Software:      Apache/2.2.22
Server Hostname:      192.168.2.134
Server Port:          80

Document Path:        /f.php
Document Length:      23 bytes

Concurrency Level:    100
Time taken for tests:  144.755 seconds
Complete requests:    10000
Failed requests:      1040
  (Connect: 0, Receive: 0, Length: 1040, Exceptions: 0)
Write errors:         0
Total transferred:    2358859 bytes
HTML transferred:     228859 bytes
Requests per second:  69.08 [#/sec] (mean)
Time per request:     1447.548 [ms] (mean)
Time per request:     14.475 [ms] (mean, across all concurrent requests)
Transfer rate:        15.91 [Kbytes/sec] received

Connection Times (ms)
      min   mean[+/-sd] median   max
Connect:    0      0  14.1      0   998
Processing:  64   1445  419.7   1396 11001
Waiting:    64   1330  431.2   1286 11001
Total:      65   1445  419.9   1397 11001

Percentage of the requests served within a certain time (ms)
 50%    1397
 66%    1471
 75%    1524
 80%    1562
 90%    1706
 95%    1930
 98%    2356
 99%    3292
100%   11001 (longest request)
javie@javi ~ $
```

4.3.4 Ab con RoundRobin Octopus

```
Terminal
Archivo  Editor  Ver  Buscar  Terminal  Ayuda

Server Port:          80

Document Path:        /f.php
Document Length:      23 bytes

Concurrency Level:    100
Time taken for tests:  58.303 seconds
Complete requests:    10000
Failed requests:      1003
  (Connect: 0, Receive: 0, Length: 1003, Exceptions: 0)
Write errors:         0
Total transferred:    2358874 bytes
HTML transferred:     228874 bytes
Requests per second:  171.52 [#/sec] (mean)
Time per request:     583.031 [ms] (mean)
Time per request:     5.830 [ms] (mean, across all concurrent requests)
Transfer rate:        39.51 [Kbytes/sec] received

Connection Times (ms)
      min   mean[+/-sd] median   max
Connect:    0      0   0.7      0    11
Processing:  14   582  493.0    451  3414
Waiting:    14   553  469.8    427  3414
Total:      15   582  493.0    451  3414

Percentage of the requests served within a certain time (ms)
 50%    451
 66%    749
 75%    890
 80%    978
 90%   1223
 95%   1483
 98%   1886
 99%   2159
100%   3414 (longest request)

Menu  Octopus - VM...  TRABAIO  Ubuntu - VMw...  Ubuntu2 - VM...  Ubuntu3 - VM...  Terminal
```

4.3.5 Ab con pesos ZenLoad

```
Terminal
Archivo Editar Ver Buscar Terminal Ayuda

Server Software:      Apache/2.2.22
Server Hostname:      192.168.2.134
Server Port:          80

Document Path:        /f.php
Document Length:      23 bytes

Concurrency Level:    100
Time taken for tests:  60.311 seconds
Complete requests:    10000
Failed requests:      974
  (Connect: 0, Receive: 0, Length: 974, Exceptions: 0)
Write errors:         0
Total transferred:    2358924 bytes
HTML transferred:     228924 bytes
Requests per second:  165.81 [#./sec] (mean)
Time per request:     603.114 [ms] (mean)
Time per request:     6.031 [ms] (mean, across all concurrent requests)
Transfer rate:        38.20 [Kbytes/sec] received

Connection Times (ms)
      min      mean[+/-sd] median    max
Connect:    0       5  69.1      0   1004
Processing: 51     597 317.5    561  2038
Waiting:    50     561 299.8    527  1991
Total:      56     602 326.4    566  2223

Percentage of the requests served within a certain time (ms)
 50%    566
 66%    785
 75%    851
 80%    897
 90%   1011
 95%   1126
 98%   1263
 99%   1402
100%   2223 (longest request)
javi@javi ~ $
```

4.3.6 Siege con menor número de conexiones Nginx

```
Terminal
Archivo Editar Ver Buscar Terminal Ayuda

HTTP/1.1 200 0.08 secs: 41 bytes ==> /f.php
HTTP/1.1 200 0.07 secs: 41 bytes ==> /f.php
HTTP/1.1 200 0.11 secs: 40 bytes ==> /f.php
HTTP/1.1 200 0.07 secs: 41 bytes ==> /f.php
HTTP/1.1 200 0.09 secs: 41 bytes ==> /f.php
HTTP/1.1 200 0.09 secs: 41 bytes ==> /f.php
HTTP/1.1 200 0.08 secs: 41 bytes ==> /f.php
HTTP/1.1 200 0.09 secs: 41 bytes ==> /f.php
HTTP/1.1 200 0.09 secs: 41 bytes ==> /f.php
HTTP/1.1 200 0.10 secs: 41 bytes ==> /f.php
HTTP/1.1 200 0.06 secs: 41 bytes ==> /f.php
HTTP/1.1 200 0.08 secs: 41 bytes ==> /f.php
HTTP/1.1 200 0.08 secs: 41 bytes ==> /f.php
HTTP/1.1 200 0.18 secs: 41 bytes ==> /f.php
HTTP/1.1 200 0.08 secs: 41 bytes ==> /f.php

Lifting the server siege...      done.
Transactions:      8852 hits
Availability:      100.00 %
Elapsed time:      59.91 secs
Data transferred:  0.35 MB
Response time:     0.10 secs
Transaction rate:  147.75 trans/sec
Throughput:        0.01 MB/sec
Concurrency:       14.98
Successful transactions: 8852
Failed transactions: 0
Longest transaction: 0.35
Shortest transaction: 0.03

FILE: /var/siege.log
You can disable this annoying message by editing
the .siegerc file in your home directory; change
the directive 'show-logfile' to false.
[error] unable to create log file: Permission denied
javi@javi ~/Escritorio/ESTUDIOIOS/3ºGrado Ingenieria Informatica/SEGUNDO CUATRIMESTRE/SERVIDORES_WEB/TRABAJO $
```

4.3.7 Siege con menor número de conexiones Octopus

```
Terminal
Archivo Editar Ver Buscar Terminal Ayuda
HTTP/1.1 200 0.11 secs: 41 bytes ==> /f.php
HTTP/1.1 200 0.07 secs: 41 bytes ==> /f.php
HTTP/1.1 200 0.08 secs: 41 bytes ==> /f.php
HTTP/1.1 200 0.08 secs: 41 bytes ==> /f.php
HTTP/1.1 200 0.08 secs: 41 bytes ==> /f.php
HTTP/1.1 200 0.10 secs: 41 bytes ==> /f.php
HTTP/1.1 200 0.08 secs: 41 bytes ==> /f.php
HTTP/1.1 200 0.08 secs: 41 bytes ==> /f.php
HTTP/1.1 200 0.07 secs: 41 bytes ==> /f.php
HTTP/1.1 200 0.07 secs: 39 bytes ==> /f.php
HTTP/1.1 200 0.10 secs: 41 bytes ==> /f.php
HTTP/1.1 200 0.07 secs: 41 bytes ==> /f.php
HTTP/1.1 200 0.13 secs: 41 bytes ==> /f.php
HTTP/1.1 200 0.06 secs: 41 bytes ==> /f.php
HTTP/1.1 200 0.07 secs: 41 bytes ==> /f.php

Lifting the server siege... done.
Transactions: 9877 hits
Availability: 99.98 %
Elapsed time: 59.84 secs
Data transferred: 0.39 MB
Response time: 0.09 secs
Transaction rate: 165.06 trans/sec
Throughput: 0.01 MB/sec
Concurrency: 14.97
Successful transactions: 9877
Failed transactions: 2
Longest transaction: 0.27
Shortest transaction: 0.03

FILE: /var/siege.log
You can disable this annoying message by editing
the .siegerc file in your home directory; change
the directive 'show-logfile' to false.
[error] unable to create log file: Permission denied
javi@javi ~ $
```

4.3.8 Siege con RoundRobin ZenLoad

```
Terminal
Archivo Editar Ver Buscar Terminal Ayuda
HTTP/1.1 200 0.20 secs: 41 bytes ==> /f.php
HTTP/1.1 200 0.15 secs: 40 bytes ==> /f.php
HTTP/1.1 200 0.18 secs: 41 bytes ==> /f.php
HTTP/1.1 200 0.21 secs: 41 bytes ==> /f.php
HTTP/1.1 200 0.17 secs: 41 bytes ==> /f.php
HTTP/1.1 200 0.27 secs: 41 bytes ==> /f.php
HTTP/1.1 200 0.23 secs: 41 bytes ==> /f.php
HTTP/1.1 200 0.23 secs: 41 bytes ==> /f.php
HTTP/1.1 200 0.17 secs: 41 bytes ==> /f.php
HTTP/1.1 200 0.19 secs: 41 bytes ==> /f.php
HTTP/1.1 200 0.19 secs: 41 bytes ==> /f.php
HTTP/1.1 200 0.14 secs: 41 bytes ==> /f.php
HTTP/1.1 200 0.15 secs: 41 bytes ==> /f.php
HTTP/1.1 200 0.20 secs: 41 bytes ==> /f.php
HTTP/1.1 200 0.24 secs: 41 bytes ==> /f.php
HTTP/1.1 200 0.15 secs: 41 bytes ==> /f.php
HTTP/1.1 200 0.25 secs: 41 bytes ==> /f.php
HTTP/1.1 200 0.22 secs: 41 bytes ==> /f.php

Lifting the server siege... done.
Transactions: 4457 hits
Availability: 100.00 %
Elapsed time: 59.84 secs
Data transferred: 0.17 MB
Response time: 0.20 secs
Transaction rate: 74.48 trans/sec
Throughput: 0.00 MB/sec
Concurrency: 14.97
Successful transactions: 4457
Failed transactions: 0
Longest transaction: 0.35
Shortest transaction: 0.10

FILE: /var/siege.log
You can disable this annoying message by editing
the .siegerc file in your home directory; change
the directive 'show-logfile' to false.
[error] unable to create log file: Permission denied
javi@javi ~ $
```

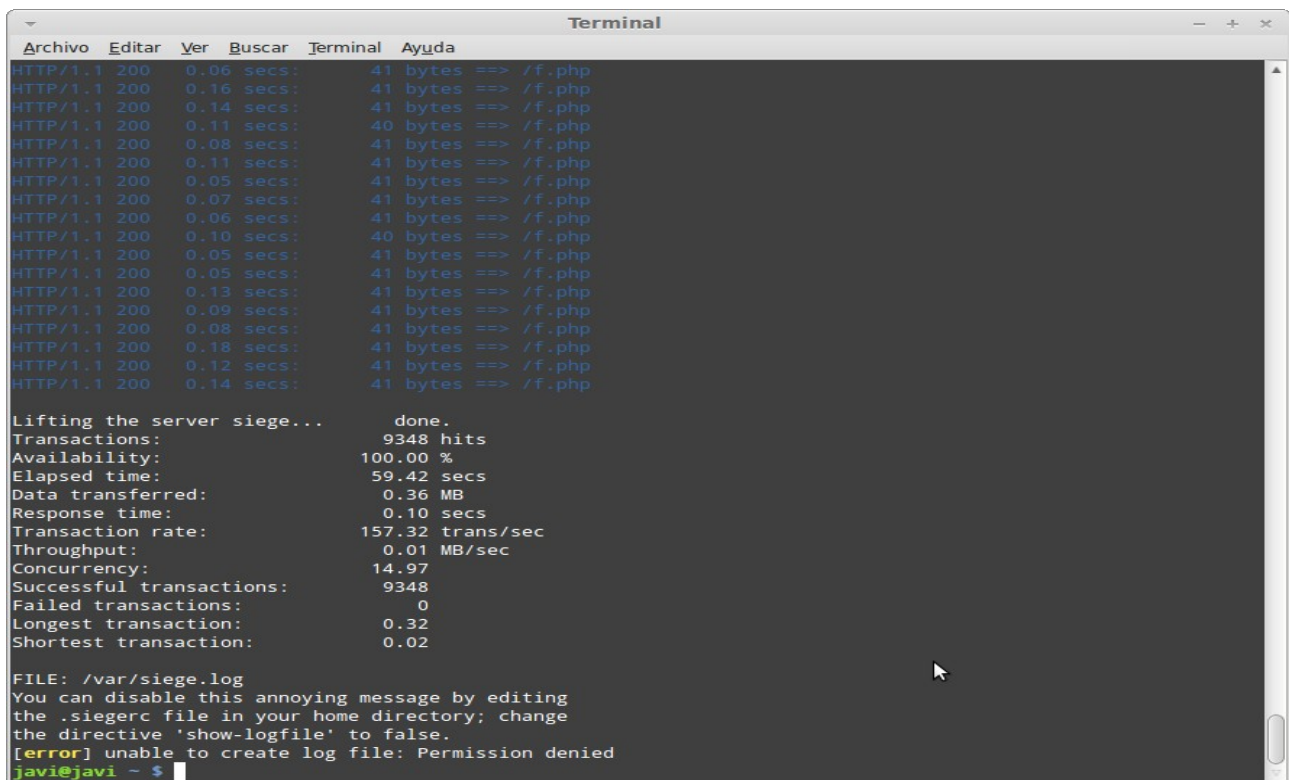

4.3.9 Siege con RoundRobin Octopus

```
Terminal
Archivo Editar Ver Buscar Terminal Ayuda
HTTP/1.1 200 0.08 secs: 41 bytes ==> /f.php
HTTP/1.1 200 0.07 secs: 41 bytes ==> /f.php
HTTP/1.1 200 0.11 secs: 41 bytes ==> /f.php
HTTP/1.1 200 0.05 secs: 41 bytes ==> /f.php
HTTP/1.1 200 0.05 secs: 41 bytes ==> /f.php
HTTP/1.1 200 0.08 secs: 41 bytes ==> /f.php
HTTP/1.1 200 0.17 secs: 41 bytes ==> /f.php
HTTP/1.1 200 0.12 secs: 41 bytes ==> /f.php
HTTP/1.1 200 0.12 secs: 41 bytes ==> /f.php
HTTP/1.1 200 0.07 secs: 41 bytes ==> /f.php
HTTP/1.1 200 0.07 secs: 41 bytes ==> /f.php
HTTP/1.1 200 0.06 secs: 41 bytes ==> /f.php
HTTP/1.1 200 0.06 secs: 41 bytes ==> /f.php
HTTP/1.1 200 0.06 secs: 41 bytes ==> /f.php
Lifting the server siege... done.
Transactions: 8876 hits
Availability: 99.91 %
Elapsed time: 59.48 secs
Data transferred: 0.35 MB
Response time: 0.09 secs
Transaction rate: 149.23 trans/sec
Throughput: 0.01 MB/sec
Concurrency: 13.72
Successful transactions: 8876
Failed transactions: 8
Longest transaction: 0.44
Shortest transaction: 0.02
FILE: /var/siege.log
You can disable this annoying message by editing
the .siegerc file in your home directory; change
the directive 'show-logfile' to false.
[error] unable to create log file: Permission denied
javi@javi ~ $
```

4.3.10 Siege con pesos Nginx

```
Terminal
Archivo Editar Ver Buscar Terminal Ayuda
HTTP/1.1 200 0.11 secs: 41 bytes ==> /f.php
HTTP/1.1 200 0.03 secs: 41 bytes ==> /f.php
HTTP/1.1 200 0.27 secs: 41 bytes ==> /f.php
HTTP/1.1 200 0.12 secs: 41 bytes ==> /f.php
HTTP/1.1 200 0.28 secs: 41 bytes ==> /f.php
HTTP/1.1 200 0.10 secs: 41 bytes ==> /f.php
HTTP/1.1 200 0.27 secs: 41 bytes ==> /f.php
HTTP/1.1 200 0.23 secs: 41 bytes ==> /f.php
HTTP/1.1 200 0.20 secs: 41 bytes ==> /f.php
HTTP/1.1 200 0.08 secs: 41 bytes ==> /f.php
HTTP/1.1 200 0.05 secs: 41 bytes ==> /f.php
HTTP/1.1 200 0.10 secs: 41 bytes ==> /f.php
HTTP/1.1 200 0.05 secs: 41 bytes ==> /f.php
HTTP/1.1 200 0.04 secs: 41 bytes ==> /f.php
HTTP/1.1 200 0.25 secs: 41 bytes ==> /f.php
Lifting the server siege... done.
Transactions: 6567 hits
Availability: 100.00 %
Elapsed time: 59.49 secs
Data transferred: 0.26 MB
Response time: 0.14 secs
Transaction rate: 110.39 trans/sec
Throughput: 0.00 MB/sec
Concurrency: 14.97
Successful transactions: 6567
Failed transactions: 0
Longest transaction: 0.60
Shortest transaction: 0.01
FILE: /var/siege.log
You can disable this annoying message by editing
the .siegerc file in your home directory; change
the directive 'show-logfile' to false.
[error] unable to create log file: Permission denied
javi@javi ~/Escritorio/ESTUDIOS/3ºGrado Ingenieria Informatica/SEGUNDO CUATRIMESTRE/SERVIDORES_WEB/TRABAJO $
```

4.3.11 Siege con pesos ZenLoad



```
Terminal
Archivo Editar Ver Buscar Terminal Ayuda
HTTP/1.1 200 0.06 secs: 41 bytes ==> /f.php
HTTP/1.1 200 0.16 secs: 41 bytes ==> /f.php
HTTP/1.1 200 0.14 secs: 41 bytes ==> /f.php
HTTP/1.1 200 0.11 secs: 40 bytes ==> /f.php
HTTP/1.1 200 0.08 secs: 41 bytes ==> /f.php
HTTP/1.1 200 0.11 secs: 41 bytes ==> /f.php
HTTP/1.1 200 0.05 secs: 41 bytes ==> /f.php
HTTP/1.1 200 0.07 secs: 41 bytes ==> /f.php
HTTP/1.1 200 0.06 secs: 41 bytes ==> /f.php
HTTP/1.1 200 0.10 secs: 40 bytes ==> /f.php
HTTP/1.1 200 0.05 secs: 41 bytes ==> /f.php
HTTP/1.1 200 0.05 secs: 41 bytes ==> /f.php
HTTP/1.1 200 0.13 secs: 41 bytes ==> /f.php
HTTP/1.1 200 0.09 secs: 41 bytes ==> /f.php
HTTP/1.1 200 0.08 secs: 41 bytes ==> /f.php
HTTP/1.1 200 0.18 secs: 41 bytes ==> /f.php
HTTP/1.1 200 0.12 secs: 41 bytes ==> /f.php
HTTP/1.1 200 0.14 secs: 41 bytes ==> /f.php

Lifting the server siege... done.
Transactions: 9348 hits
Availability: 100.00 %
Elapsed time: 59.42 secs
Data transferred: 0.36 MB
Response time: 0.10 secs
Transaction rate: 157.32 trans/sec
Throughput: 0.01 MB/sec
Concurrency: 14.97
Successful transactions: 9348
Failed transactions: 0
Longest transaction: 0.32
Shortest transaction: 0.02

FILE: /var/siege.log
You can disable this annoying message by editing
the .siegerc file in your home directory; change
the directive 'show-logfile' to false.
[error] unable to create log file: Permission denied
javi@javi ~ $
```

5. Bibliografía

ZenLoadBalance – www.zenloadbalancer.com

Nginx- <http://nginx.org>

Octopus- <http://sourceforge.net/projects/octopuslb/>

Documentación de la asignatura de Servidores Web de Altas Prestaciones de la UGR.