

## **Práctica 3:**

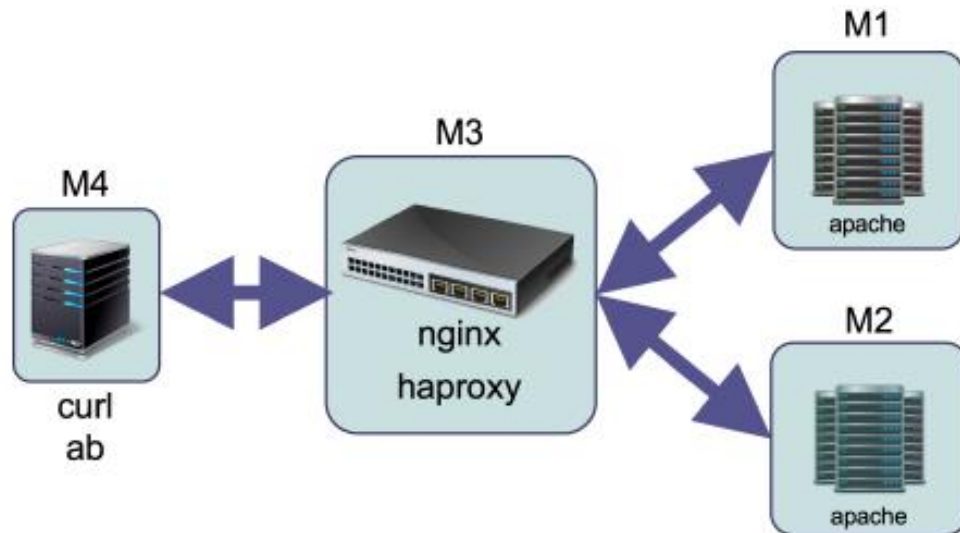
### **Balanceo de carga en un sitio web**

#### ***Índice:***

- 1) Introducción*
- 2) Creación y configuración del balanceador*
- 3) Balanceo de carga con Nginx*
- 4) Balanceo de carga con HAProxy*
- 5) Balanceo de carga con Pound (optativo)*
- 6) Benchmark-ab para someter a una alta carga a la granja web*
- 7) Conclusión*

## 1) Introducción

En esta práctica configuraremos una red entre varias máquinas de forma que tengamos un balanceador que reparta la carga entre varios servidores finales para solucionar el problema de la sobrecarga de los servidores ante peticiones web HTTP. Así conseguiremos una infraestructura redundante y de alta disponibilidad.



Deberemos tener en ejecución las máquinas servidoras finales m1 y m2 ya configuradas con el servidor Apache.

Crearemos una tercera máquina m3 en la que no debe haber ningún software que se apropie del puerto 80, por lo que nos aseguraremos de que no esté instalado el Apache. Este puerto lo necesitará el software de balanceo para recibir las peticiones HTTP desde fuera de la granja web.

Las direcciones IP de las tres máquinas serán:

-m1 → 192.168.56.101

-m2 → 192.168.56.102

-m3 → 192.168.56.103

Una vez instalemos y configuremos el software de balanceo (Nginx, HAProxy y/o Pound), desde una máquina externa a la granja web (será un terminal en el ordenador anfitrión) ejecutaremos la herramienta curl para hacer peticiones HTTP a la IP de la máquina balanceadora (la VIP de nuestro sistema).

Finalmente realizaremos una serie de benchmarks para comparar los distintos softwares de balanceo utilizados y analizar cómo se comportan ante una alta carga de peticiones web.



### 3) Balanceo de carga con Nginx

Nginx es un servidor web ligero de alto rendimiento, usado por: WordPress, Hulu, GitHub, Ohloh, SourceForge, TorrentReactor y un largo etcétera.

#### 3.1) Instalación Nginx

Para instalarlo ejecutamos el siguiente comando en la máquina m3:

```
sudo apt-get update && sudo apt-get dist-upgrade && sudo apt-get autoremove  
sudo apt-get install nginx  
sudo systemctl start nginx
```

```
joselepedraza@m3:~$ sudo apt-get install nginx  
Leyendo lista de paquetes... Hecho  
Creando árbol de dependencias  
Leyendo la información de estado... Hecho  
Se instalarán los siguientes paquetes adicionales:  
fontconfig-config fonts-dejavu-core libfontconfig1 libgd3 libjpeg-turbo8 libjpeg8  
libnginx-mod-http-geoip libnginx-mod-http-image-filter libnginx-mod-http-xslt-filter  
libnginx-mod-mail libnginx-mod-stream libtiff5 libwebp6 libxpm4 nginx-common nginx-core  
Paquetes sugeridos:  
libgd-tools fcgiwrap nginx-doc ssl-cert  
Se instalarán los siguientes paquetes NUEVOS:  
fontconfig-config fonts-dejavu-core libfontconfig1 libgd3 libjpeg-turbo8 libjpeg8  
libnginx-mod-http-geoip libnginx-mod-http-image-filter libnginx-mod-http-xslt-filter  
libnginx-mod-mail libnginx-mod-stream libtiff5 libwebp6 libxpm4 nginx nginx-common nginx-core  
0 actualizados, 18 nuevos se instalarán, 0 para eliminar y 0 no actualizados.  
Se necesita descargar 2.461 kB de archivos.  
Se utilizarán 8.210 kB de espacio de disco adicional después de esta operación.  
¿Desea continuar? [S/n] s  
Des:1 http://es.archive.ubuntu.com/ubuntu bionic-updates/main amd64 libjpeg-turbo8 amd64 1.5.2-0ubuntu5.18.04.3 [110 kB]  
Des:2 http://es.archive.ubuntu.com/ubuntu bionic/main amd64 fonts-dejavu-core all 2.37-1 [1.041 kB]  
Des:3 http://es.archive.ubuntu.com/ubuntu bionic/main amd64 fontconfig-config all 2.12.6-0ubuntu2 [55,8 kB]  
Des:4 http://es.archive.ubuntu.com/ubuntu bionic/main amd64 libfontconfig1 amd64 2.12.6-0ubuntu2 [137 kB]  
Des:5 http://es.archive.ubuntu.com/ubuntu bionic/main amd64 libjpeg8 amd64 8c-2ubuntu8 [2.194 B]  
Des:6 http://es.archive.ubuntu.com/ubuntu bionic/main amd64 libjpeg-turbo8 amd64 2.1-3.1build1 [26,7 kB]  
Des:7 http://es.archive.ubuntu.com/ubuntu bionic-updates/main amd64 libtiff5 amd64 4.0.9-5ubuntu0.3 [153 kB]  
Des:8 http://es.archive.ubuntu.com/ubuntu bionic/main amd64 libwebp6 amd64 0.6.1-2 [185 kB]  
Des:9 http://es.archive.ubuntu.com/ubuntu bionic/main amd64 libxpm4 amd64 1:3.5.12-1 [34,0 kB]  
Des:10 http://es.archive.ubuntu.com/ubuntu bionic-updates/main amd64 libgd3 amd64 2.2.5-4ubuntu0.3 [119 kB]  
Des:11 http://es.archive.ubuntu.com/ubuntu bionic-updates/main amd64 nginx-common all 1.14.0-0ubuntu
```

Comprobamos la versión con `nginx -v`:

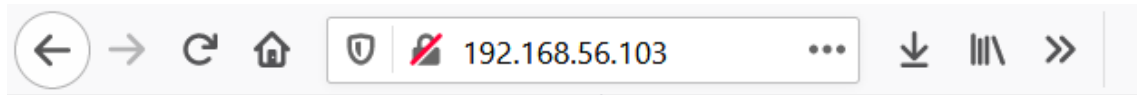
```
joselepedraza@m3:~$ nginx -v  
nginx version: nginx/1.14.0 (Ubuntu)  
joselepedraza@m3:~$ _
```

Tutorial detallado de la instalación en:

<https://www.liberiangeek.net/2016/07/how-to-install-nginx-webserver-on-ubuntu-16-04/>

Servidores Web de Altas Prestaciones

Para comprobar que está funcionando correctamente accedemos desde el navegador de nuestro equipo a la dirección VIP:



# Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to [nginx.org](https://nginx.org).  
Commercial support is available at [nginx.com](https://nginx.com).

*Thank you for using nginx.*

## 3.2) Configuración balanceo de carga con Nginx

Nginx soporta la realización de balanceo de carga mediante la directiva `proxy_pass`. En nuestro caso nos interesa redirigir el tráfico a un grupo de servidores. Para definir este grupo deberemos dar un nombre al conjunto mediante la directiva `upstream`.

Lo primero que debemos hacer es abrir el archivo de configuración por defecto de Nginx con el siguiente comando (si no existe lo creamos):

```
sudo vim /etc/nginx/conf.d/default.conf
```

```

upstream backendApache{
    server 192.168.56.101;
    server 192.168.56.102;
}

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://backendApache;
        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 "";
    }
}

```

Primero, definimos qué máquinas formarán el cluster web (backend) y en qué puertos está a la escucha el servidor web correspondiente, en nuestro caso, los servidores apache de m1 y m2. Esto se especifica en la sección “upstream” de la configuración, en la cual vamos a indicar las IP de todos los servidores finales de nuestra granja web.

A continuación de esta sección debemos indicarle al Nginx que use ese grupo definido en el “upstream” como las máquinas a donde se repartirá el tráfico. Esto se especifica en la sección “server” (es importante definir el “upstream” al principio del fichero, antes y fuera de la sección “server”). También es importante para que el proxy\_pass funcione correctamente que indiquemos que la conexión entre Nginx y los servidores finales será HTTP 1.1 así como especificarle que debe eliminar la cabecera Connection (hacerla vacía con “”) para evitar que se pase al servidor final la cabecera que indica el usuario.

En este ejemplo usamos el balanceo mediante el algoritmo de “round-robin” por turnos con la misma prioridad para todos los servidores.

Además, debemos indicar en el archivo de configuración de Nginx que use nuestra configuración anteriormente descrita y que Nginx no actúe como servidor y solo lo haga como balanceador. Para esto ejecutamos el siguiente comando:

```
sudo vim /etc/nginx/nginx.conf
```

Y comentamos la línea “*#include /etc/nginx/sites-enabled/\**”.

```

##
# Gzip Settings
##

gzip on;

# gzip_vary on;
# gzip_proxied any;
# gzip_comp_level 6;
# gzip_buffers 16 8k;
# gzip_http_version 1.1;
# gzip_types text/plain text/css application/json application/javascript text/xml applicatio
n/xml application/xml+rss text/javascript;

##
# Virtual Host Configs
##

include /etc/nginx/conf.d/*.conf;
#include /etc/nginx/sites-enabled/*;
}

#mail {
#   # See sample authentication script at:
#   # http://wiki.nginx.org/ImapAuthenticateWithApachePhpScript
#
#   # auth_http localhost/auth.php;
#   # pop3_capabilities "TOP" "USER";
#   # imap_capabilities "IMAP4rev1" "UIDPLUS";
#
#   server {
#       listen     localhost:110;
#       protocol   pop3;
#       proxy      on;
#   }
#}
"/etc/nginx/nginx.conf" 85L, 1483C escritos
joselepedraza@m3:~$ _

```

Una vez con esta configuración, relanzamos el servicio Nginx con el comando:

*sudo service nginx restart*

```

joselepedraza@m3:~$ sudo service nginx restart
joselepedraza@m3:~$ _

```

Como no obtenemos ningún mensaje de error, todo está funcionando correctamente y ya podemos probar la configuración haciendo peticiones a la IP de esta máquina m3 que actúa como balanceador de carga entre el usuario y el backend. Por ejemplo, podemos usar el comando cURL como sigue:

*curl <http://192.168.56.103/ejemplo.html>*

*curl <http://192.168.56.103/ejemplo.html>*

Esto debería mostrar la página de inicio de cada una de las máquinas, alternativamente, lo que querrá decir que está repartiendo las peticiones entre ambos (entre m1 y m2). Para ello deberemos modificar el .html a modo de prueba para asegurarnos de su correcto funcionamiento (a cada hora ambas máquinas tendrán los .html idénticos, ya que tenemos programada la réplica de m1 en m2 cada hora).

```
C:\Users\Josele>curl 192.168.56.103/ejemplo.html
<html>
  <body>
    Web de ejemplo de joselepdraza para SWAP(m1)
  </body>
</html>

C:\Users\Josele>curl 192.168.56.103/ejemplo.html
<html>
  <body>
    Web de ejemplo de joselepdraza para SWAP(m2)
  </body>
</html>

C:\Users\Josele>curl 192.168.56.103/ejemplo.html
<html>
  <body>
    Web de ejemplo de joselepdraza para SWAP(m1)
  </body>
</html>

C:\Users\Josele>curl 192.168.56.103/ejemplo.html
<html>
  <body>
    Web de ejemplo de joselepdraza para SWAP(m2)
  </body>
</html>

C:\Users\Josele>
```

Funciona correctamente, distribuyendo la carga por turnos.

### 3.2.2) *Otras configuraciones de balanceo con Nginx*

#### **Ponderación (pesos):**

En este caso, en vez de llevar una distribución de carga por turnos round-robin, asignaremos un peso a cada servidor de producción, de manera que, de cada 3 peticiones que lleguen al balanceador (VIP: 192.168.56.103), la máquina m2 (192.168.56.102) atenderá 2 y la máquina m1 atenderá 1 (192.168.56.101).

Lo indicaremos en la sección “upstream” del fichero `/etc/nginx/conf.d/default.conf` añadiendo la opción *weight* a cada línea como vemos a continuación:



```
upstream backendApache{
    server 192.168.56.101 weight=1;
    server 192.168.56.102 weight=2;
}

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://backendApache;
        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 "";
    }
}
```

Comprobamos desde el símbolo de sistema de mi equipo que funciona correctamente, no olvidar relanzar el servicio con:

*sudo service nginx restart*

```
C:\Users\Josele>curl 192.168.56.103/ejemplo.html
<html>
  <body>
    Web de ejemplo de joselepdraza para SWAP(m1)
  </body>
</html>

C:\Users\Josele>curl 192.168.56.103/ejemplo.html
<html>
  <body>
    Web de ejemplo de joselepdraza para SWAP(m2)
  </body>
</html>

C:\Users\Josele>curl 192.168.56.103/ejemplo.html
<html>
  <body>
    Web de ejemplo de joselepdraza para SWAP(m2)
  </body>
</html>

C:\Users\Josele>curl 192.168.56.103/ejemplo.html
<html>
  <body>
    Web de ejemplo de joselepdraza para SWAP(m1)
  </body>
</html>

C:\Users\Josele>
```

Como vemos hace correctamente el reparto según los pesos asignados.

### Peticiones de misma IP a mismo servidor

En este caso, nos interesa repartir la carga de peticiones provenientes de la misma IP al mismo servidor (por ejemplo, el caso del carrito de la compra). Con el uso de “ip\_hash” en la sección “upstream” del fichero de configuración conseguiremos que todas las peticiones que vengan de la misma IP se dirijan a la misma máquina servidora final.

Esto tiene un inconveniente, y es que todos los usuarios detrás de un proxy o NAT son dirigidos al mismo backend, por lo que pueden producirse balanceos no equilibrados. Para evitarlo, haremos uso de “keepalive” en la sección “upstream”, lo cual mantiene la conexión entre balanceador y servidor final durante un tiempo limitado (en segundos).

Resetamos el servicio de Nginx y comprobamos que todas las peticiones realizadas desde la IP de mi máquina las atenderá la máquina m2:

```
C:\Users\Josele>curl 192.168.56.103/ejemplo.html
<html>
    <body>
        Web de ejemplo de joselepdraza para SWAP(m2)
    </body>
</html>

C:\Users\Josele>curl 192.168.56.103/ejemplo.html
<html>
    <body>
        Web de ejemplo de joselepdraza para SWAP(m2)
    </body>
</html>

C:\Users\Josele>curl 192.168.56.103/ejemplo.html
<html>
    <body>
        Web de ejemplo de joselepdraza para SWAP(m2)
    </body>
</html>

C:\Users\Josele>curl 192.168.56.103/ejemplo.html
<html>
    <body>
        Web de ejemplo de joselepdraza para SWAP(m2)
    </body>
</html>
```

Otras opciones de configuración en “upstream”:

- *weight* = number → Permite especificar un peso para el servidor, por defecto 1.
- *max\_fails* = number → Especifica un numero de intentos de comunicación erróneos en “fail\_timeout” segundos para considerar al servidor no operativo (por defecto 1).
- *fail\_timeout* = time → Indica el tiempo en el que deben ocurrir “max\_fails” intentos fallidos de conexión para considerar al servidor no operativo. Por defecto es 10 segundos.
- *down* → Marca el servidor como permanentemente offline (para ser usado con *ip\_hash*). Útil para tareas de mantenimiento por ejemplo.
- *backup* → Reserva este servidor y solo le pasa tráfico si alguno de los otros servidores no-backup está caído u ocupado. No compatible con la directiva “ip\_hash”.

Servidores Web de Altas Prestaciones

## Backup en m2

En este caso, establecemos que la máquina m2 actúe como backup de la máquina m1 cuando ésta no esté disponible:

```
upstream backendApache{
    server 192.168.56.101;
    server 192.168.56.102 backup;
}

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://backendApache;
        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 "";
    }
}
```

Comprobamos su funcionamiento apagando la máquina m1 después de ejecutar varios cURLs:

```
C:\Users\Josele>curl 192.168.56.103/ejemplo.html
<html>
    <body>
        Web de ejemplo de joselepdrza para SWAP(m1)
    </body>
</html>

C:\Users\Josele>curl 192.168.56.103/ejemplo.html
<html>
    <body>
        Web de ejemplo de joselepdrza para SWAP(m1)
    </body>
</html>

C:\Users\Josele>curl 192.168.56.103/ejemplo.html
<html>
    <body>
        Web de ejemplo de joselepdrza para SWAP(m1)
    </body>
</html>

C:\Users\Josele>curl 192.168.56.103/ejemplo.html
<html>
    <body>
        Web de ejemplo de joselepdrza para SWAP(m2)
    </body>
</html>
```

## 4) Balanceo de carga con HAProxy

HAProxy es un balanceador de carga y también proxy, de forma que puede balancear cualquier tipo de tráfico. Es un software muy adecuado para repartir carga y construir una infraestructura de altas prestaciones, con una configuración básica y sencilla, posee muchas opciones para realizar cualquier tipo de balanceo y control de los servidores finales.

Antes de proceder con la instalación y configuración de HAProxy, debemos parar el servicio de Nginx dado que no pueden estar ambos balanceadores por el mismo puerto (80) y los dos a la vez no podrán estar funcionando. Ejecutamos:

```
sudo service nginx stop      (o en otro caso: sudo service haproxy stop)
```

### 4.1) Instalación HAProxy

Para instalar la herramienta basta con ejecutar:

```
sudo apt-get install haproxy
```

```
joselepedraza@m3:~$ sudo apt-get install haproxy
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias
Leyendo la información de estado... Hecho
Se instalarán los siguientes paquetes adicionales:
  liblua5.3-0
Paquetes sugeridos:
  vim-haproxy haproxy-doc
Se instalarán los siguientes paquetes NUEVOS:
  haproxy liblua5.3-0
0 actualizados, 2 nuevos se instalarán, 0 para eliminar y 0 no actualizados.
Se necesita descargar 1.231 kB de archivos.
Se utilizarán 2.842 kB de espacio de disco adicional después de esta operación.
¿Desea continuar? [S/n] S
Des:1 http://es.archive.ubuntu.com/ubuntu bionic-updates/main amd64 liblua5.3-0 amd64 5.3.3-1ubuntu0
.18.04.1 [115 kB]
Des:2 http://es.archive.ubuntu.com/ubuntu bionic-updates/main amd64 haproxy amd64 1.8.8-1ubuntu0.9 [
1.117 kB]
Descargados 1.231 kB en 1s (1.093 kB/s)
Seleccionando el paquete liblua5.3-0:amd64 previamente no seleccionado.
(Leyendo la base de datos ... 67243 ficheros o directorios instalados actualmente.)
Preparando para desempaquetar .../liblua5.3-0_5.3.3-1ubuntu0.18.04.1_amd64.deb ...
Desempaquetando liblua5.3-0:amd64 (5.3.3-1ubuntu0.18.04.1) ...
Seleccionando el paquete haproxy previamente no seleccionado.
Preparando para desempaquetar .../haproxy_1.8.8-1ubuntu0.9_amd64.deb ...
Desempaquetando haproxy (1.8.8-1ubuntu0.9) ...
Configurando liblua5.3-0:amd64 (5.3.3-1ubuntu0.18.04.1) ...
Configurando haproxy (1.8.8-1ubuntu0.9) ...
Created symlink /etc/systemd/system/multi-user.target.wants/haproxy.service → /lib/systemd/system/ha
proxy.service.
Procesando disparadores para libc-bin (2.27-3ubuntu1) ...
Procesando disparadores para systemd (237-3ubuntu10.39) ...
Procesando disparadores para man-db (2.8.3-2ubuntu0.1) ...
Procesando disparadores para rsyslog (8.32.0-1ubuntu4) ...
Procesando disparadores para ureadahead (0.100.0-21) ...
joselepedraza@m3:~$
```

Para iniciar el servicio basta con ejecutar:

```
sudo systemctl start haproxy
```

ó

```
sudo service haproxy start
```

Servidores Web de Altas Prestaciones

## 4.2) Configuración balanceo de carga con HAProxy

Lo primero que deberemos hacer es abrir el archivo de configuración por defecto de HAProxy, para indicarle cuáles son nuestros servidores (backend) y qué peticiones balancear, con el siguiente comando (si no existe lo creamos):

```
sudo vim /etc/haproxy/haproxy.cfg
```

Un balanceador sencillo debe escuchar tráfico en el puerto 80 y redirigirlo a alguna de las máquinas servidoras finales, m1 o m2 en nuestro caso. Usaremos la configuración siguiente:

```
# Default SSL material locations
ca-base /etc/ssl/certs
crt-base /etc/ssl/private

# Default ciphers to use on SSL-enabled listening sockets.
# For more information, see ciphers(1SSL). This list is from:
# https://hynek.me/articles/hardening-your-web-servers-ssl-ciphers/
# An alternative list with additional directives can be obtained from
# https://mozilla.github.io/server-side-tls/ssl-config-generator/?server=haproxy
ssl-default-bind-ciphers ECDH+AESGCM:DH+AESGCM:ECDH+AES256:DH+AES256:ECDH+AES128:DH+AES:RSA+
AESGCM:RSA+AES:!aNULL:!MD5:!DSS
ssl-default-bind-options no-sslv3

defaults
    log         global
    mode        http
    option      httplog
    option      dontlognull
    timeout connect 5000
    timeout client 50000
    timeout server 50000
    errorfile 400 /etc/haproxy/errors/400.http
    errorfile 403 /etc/haproxy/errors/403.http
    errorfile 408 /etc/haproxy/errors/408.http
    errorfile 500 /etc/haproxy/errors/500.http
    errorfile 502 /etc/haproxy/errors/502.http
    errorfile 503 /etc/haproxy/errors/503.http
    errorfile 504 /etc/haproxy/errors/504.http

frontend http-in
    bind *:80
    default_backend servidoresSWAP

backend servidoresSWAP
    server m1 192.168.56.101:80 maxconn 32
    server m2 192.168.56.102:80 maxconn 32
```

Una vez salvada la configuración en el fichero, lanzamos el servicio HAProxy mediante el comando:

```
sudo /usr/sbin/haproxy -f /etc/haproxy/haproxy.cfg
```

ó

```
sudo service haproxy restart
```

Podemos ver configuraciones no tan básicas, pero igual de funcionales en el manual:

<http://www.haproxy.org/download/1.4/doc/configuration.txt>

Comprobamos que funciona correctamente con la configuración aplicada:

```
C:\Users\Josele>curl 192.168.56.103/ejemplo.html
<html>
    <body>
        Web de ejemplo de joselepedraza para SWAP(m1)
    </body>
</html>

C:\Users\Josele>curl 192.168.56.103/ejemplo.html
<html>
    <body>
        Web de ejemplo de joselepedraza para SWAP(m2)
    </body>
</html>

C:\Users\Josele>curl 192.168.56.103/ejemplo.html
<html>
    <body>
        Web de ejemplo de joselepedraza para SWAP(m2)
    </body>
</html>

C:\Users\Josele>curl 192.168.56.103/ejemplo.html
<html>
    <body>
        Web de ejemplo de joselepedraza para SWAP(m1)
    </body>
</html>
```

Como vemos, muestra la página buscada con cURL de cada una de las máquinas, alternativamente, lo que quiere decir que está repartiendo correctamente las peticiones entre ambas.

Para detener el servicio de HAProxy me he dado cuenta de que no basta con ejecutar el comando anteriormente indicado: *sudo service haproxy stop*

Por lo que deberemos matar el proceso de la siguiente manera cuando queramos desactivar el servicio:

*sudo pkill haproxy*

```
joselepedraza@m3:~$ sudo service haproxy stop
joselepedraza@m3:~$ sudo netstat -tulpn | grep :80
tcp        0      0 0.0.0.0:80          0.0.0.0:*          LISTEN      2819/haproxy
joselepedraza@m3:~$ sudo pkill haproxy
joselepedraza@m3:~$ sudo netstat -tulpn | grep :80
joselepedraza@m3:~$ _
```

## 5) Balanceo de carga con Pound (optativo)

El objetivo de Pound es poder redirigir una petición de entrada entre varios servidores, lo que llamamos balanceador de carga, como hemos visto hasta ahora. Pero otra función muy resaltable de Pound, es la capacidad de recibir conexiones https y enviarlas por http a nuestro servidor final. De esta forma la conexión entre el cliente y nuestro balanceador está encriptada, y desde este hasta el servidor final (en nuestro caso web) va en texto claro ya que estamos dentro de nuestro entorno “privado”.

### 5.1) Instalación Pound

Para instalar la herramienta deberemos tener previamente Apache2 instalado previamente, por lo que ejecutaremos lo siguiente:

```
sudo apt-get install apache2
```

En este caso, la instalación de Pound no se puede hacer ejecutando directamente el comando: *sudo apt-get install pound*, ya que Pound no está en los repositorios online de Ubuntu, por lo que tendremos el siguiente error:

```
joselepedraza@m3:~$ sudo apt-get install pound
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias
Leyendo la información de estado... Hecho
El paquete pound no está disponible, pero algún otro paquete hace referencia
a él. Esto puede significar que el paquete falta, está obsoleto o sólo se
encuentra disponible desde alguna otra fuente

E: El paquete «pound» no tiene un candidato para la instalación
joselepedraza@m3:~$ _
```

Por lo que tendremos que proceder a instalarlo manualmente descargando el paquete .deb y procediendo a la instalación como sigue:

Link: [launchpadlibrarian.net/317629201/pound\\_2.7-1.3\\_amd64.deb](http://launchpadlibrarian.net/317629201/pound_2.7-1.3_amd64.deb)

Descargamos desde el link de descarga anterior:

```
joselepedraza@m3:~$ curl -O launchpadlibrarian.net/317629201/pound_2.7-1.3_amd64.deb
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left   Speed
100 95406  100 95406    0     0   235k      0  0:00:00  0:00:00  0:00:00  235k
```



Una vez descargado, vemos donde se encuentra el paquete, lo descomprimimos, instalamos y comprobamos el estado de Pound como muestro a continuación:

```
joselepedraza@m3:~$ ls
pound_2.7-1.3_amd64.deb
joselepedraza@m3:~$ sudo dpkg -i pound_2.7-1.3_amd64.deb
Seleccionando el paquete pound previamente no seleccionado.
(Leyendo la base de datos ... 68082 ficheros o directorios instalados actualmente.)
Preparando para desempaquetar pound_2.7-1.3_amd64.deb ...
Desempaquetando pound (2.7-1.3) ...
Configurando pound (2.7-1.3) ...
Procesando disparadores para systemd (237-3ubuntu10.39) ...
Procesando disparadores para ureadahead (0.100.0-21) ...
Procesando disparadores para man-db (2.8.3-2ubuntu0.1) ...
joselepedraza@m3:~$ sudo service pound status
● pound.service - LSB: reverse proxy and load balancer
   Loaded: loaded (/etc/init.d/pound; generated)
   Active: active (exited) since Mon 2020-03-30 13:16:18 UTC; 43s ago
     Docs: man:systemd-sysv-generator(8)
    Tasks: 0 (limit: 1108)
   CGroup: /system.slice/pound.service

mar 30 13:16:18 m3 systemd[1]: Starting LSB: reverse proxy and load balancer...
mar 30 13:16:18 m3 pound[1999]: * pound will not start unconfigured.
mar 30 13:16:18 m3 pound[1999]: * Please configure; afterwards, set startup=1 in /etc/default/pound
mar 30 13:16:18 m3 systemd[1]: Started LSB: reverse proxy and load balancer.
lines 1-11/11 (END)
```

Como vemos ya está instalado y no se iniciará el servicio hasta que no esté configurado.

## 5.2) Configuración balanceo de carga con Pound

Lo primero que haré será crear una copia de seguridad del archivo de configuración de Pound de la siguiente manera:

```
sudo cp /etc/pound/pound.cfg /etc/pound/pound.cfg.old
```

Una vez hecho esto procedemos a editar dicho archivo con *vim* como vemos:

```
##      1      normal
##      2      extended
##      3      Apache-style (common log format)
LogLevel      1

## check backend every X secs:
Alive          30

## use hardware-acceleration card supported by openssl(1):
#SSLEngine     "<hw>"

# poundctl control socket
Control "/var/run/pound/poundctl.socket"

#####
## listen, redirect and ... to:

## redirect all requests on port 8080 ("ListenHTTP") to the local webserver (see "Service" below):
ListenHTTP
    Address 192.168.56.103
    Port    80
End
Service
    BackEnd
        Address 192.168.56.101
        Port    80
        Priority 1
    End
    BackEnd
        Address 192.168.56.102
        Port    80
        Priority 1
    End
End
"/etc/pound/pound.cfg" 49L, 924C escritos
```

Según esta configuración, se está usando el puerto 80 de nuestras dos máquinas (m1 y m2) y también escucha por el puerto 80 del propio balanceador (m3).

Para iniciar el servicio de Pound primero debemos habilitarlo, cambiando en el siguiente archivo la línea de *startup=0* por *startup=1*, ejecutamos:

```
sudo vim /etc/default/pound
```

```
# Defaults for pound initscript
# sourced by /etc/init.d/pound
# installed at /etc/default/pound by the maintainer scripts

# prevent startup with default configuration
# set the below variable to 1 in order to allow pound to start
startup=1
```

Finalmente, para iniciar el servicio usamos:

```
sudo /etc/init.d/pound start
```

y

```
sudo service pound restart
```

```
joselepedraza@m3:~$ sudo /etc/init.d/pound start
[ ok ] Starting pound (via systemctl): pound.service.
joselepedraza@m3:~$ _
```

Para pararlo usaremos:

```
sudo /etc/init.d/pound stop
```

Con el siguiente comando comprobamos que esto último funciona como lo hemos configurado:

```
sudo netstat -tulpn | grep :80
```

```
joselepedraza@m3:~$ sudo service pound restart
joselepedraza@m3:~$ sudo netstat -tulpn | grep :80
tcp        0      0 192.168.56.103:80      0.0.0.0:*               LISTEN      2689/pound
joselepedraza@m3:~$
```

Procedemos a comprobar su funcionamiento desde el terminal de Windows de mi máquina anfitrión mandándole unas cuantas peticiones al balanceador (m3):

```
C:\Users\Josele>curl 192.168.56.103/ejemplo.html
<html>
  <body>
    Web de ejemplo de joselepedraza para SWAP(m1)
  </body>
</html>

C:\Users\Josele>curl 192.168.56.103/ejemplo.html
<html>
  <body>
    Web de ejemplo de joselepedraza para SWAP(m2)
  </body>
</html>

C:\Users\Josele>curl 192.168.56.103/ejemplo.html
<html>
  <body>
    Web de ejemplo de joselepedraza para SWAP(m2)
  </body>
</html>

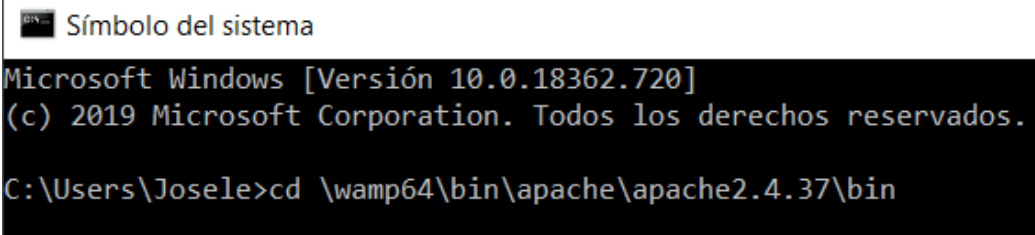
C:\Users\Josele>curl 192.168.56.103/ejemplo.html
<html>
  <body>
    Web de ejemplo de joselepedraza para SWAP(m1)
  </body>
</html>
```

Funciona correctamente, aunque observamos que, por defecto, Pound no utiliza Round-Robin como algoritmo de balanceo como si lo hacen Nginx y HAProxy.

## 6) Benchmark-ab para someter a una alta carga a la granja web

En este punto vamos a medir el rendimiento del servidor y qué herramienta actúa (bajo las mismas condiciones o similares) mejor como balanceador de carga.

Para ello, en la máquina anfitrión Windows debemos tener instalado Apache, en mi caso tengo hecha la instalación con WampServer, por lo que deberemos acceder a la ruta siguiente:



```
Símbolo del sistema
Microsoft Windows [Versión 10.0.18362.720]
(c) 2019 Microsoft Corporation. Todos los derechos reservados.
C:\Users\Josele>cd \wamp64\bin\apache\apache2.4.37\bin
```

Una vez aquí ya podemos ejecutar el Benchmark de Apache llamado “ab”, que es una utilidad que se instala junto al servidor Apache y permite comprobar el rendimiento de cualquier servidor web. Ejecutamos el comando “ab” como sigue (enviamos la carga al balanceador de la máquina m3, VIP):

```
ab -n 10000 -c 10 http://192.168.56.103/ejemplo.html
```

Usaremos la siguiente orden para guardar los resultados en un fichero .csv para poder graficarlos posteriormente:

```
ab -g resultados.csv -n 10000 -c 10 http://192.168.56.103/ejemplo.html
```

Los parámetros indicados en la orden anterior le indican al benchmark que solicite la página con dirección <http://192.168.56.103/ejemplo.html> 10000 veces (-n 10000 indica el número de peticiones) y hacer esas peticiones concurrentemente de 10 en 10 (-c 10 indica el nivel de concurrencia).

Si mientras la máquina que ejecuta la carga (Windows), ejecutamos un *top* en todas las máquinas (m1, m2, m3) podremos apreciar cómo afecta el nivel de peticiones a cada elemento de nuestra granja web.

Procedemos a hacer las mediciones con las diferentes herramientas vistas en la práctica (ejecutamos 3 veces el benchmark y nos quedamos con la mediana en cada caso).

## - Nginx:

m1 [Corriendo] - Oracle VM VirtualBox

Archivo Máquina Ver Entrada Dispositivos Ayuda

top - 22:22:14 up 2:25, 1 user, load average: 0.24, 0.05, 0.02  
Tasks: 69 total, 2 running, 48 sleeping, 0 stopped, 0 zombie  
%Cpu(s): 11.8 us, 16.7 sy, 0.0 ni, 56.5 id, 0.0 wa, 0.0 hi, 15.0 si, 0.0 st  
KiB Mem : 1008804 total, 167636 free, 284648 used, 556520 buff/cache  
KiB Swap: 2017276 total, 2017276 free, 0 used, 579228 avail Mem

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
894	www-data	20	0	826812	5668	3536	S	19.3	0.6	0:15.01	apache2
893	www-data	20	0	826812	5664	3532	S	19.3	0.6	0:15.03	apache2
7	root	20	0	0	0	0	R	1.0	0.0	0:01.49	ksoftirqd/0
8	root	20	0	0	0	0	I	0.3	0.0	0:00.84	rcu_sched
1606	joselep+	20	0	42672	3780	3200	R	0.3	0.4	0:10.95	top
1	root	20	0	77940	8916	6556	S	0.0	0.9	0:01.88	systemd
2	root	20	0	0	0	0	S	0.0	0.0	0:00.58	kthreadd
4	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworke/0:0H
6	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	mm_percpu_wq

Símbolo del sistema - ab -g resNginx.csv -n 10000 -c 10 http://192.168.56.103/ejemplo.html

C:\wamp64\bin\apache\apache2.4.37\bin>ab -g resNginx.csv -n 10000 -c 10 http://192.168.56.103/ejemplo.html

This is ApacheBench, Version 2.3 <\$Revision: 1843412 \$>  
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/  
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking 192.168.56.103 (be patient)  
Completed 1000 requests  
Completed 2000 requests  
Completed 3000 requests  
Completed 4000 requests  
Completed 5000 requests  
Completed 6000 requests  
Completed 7000 requests  
Completed 8000 requests

m2 [Corriendo] - Oracle VM VirtualBox

Archivo Máquina Ver Entrada Dispositivos Ayuda

top - 22:22:16 up 2:25, 1 user, load average: 0.02, 0.02, 0.00  
Tasks: 89 total, 2 running, 48 sleeping, 0 stopped, 0 zombie  
%Cpu(s): 15.0 us, 19.4 sy, 0.0 ni, 46.9 id, 0.0 wa, 0.0 hi, 18.7 si, 0.0 st  
KiB Mem : 1008804 total, 157460 free, 284664 used, 566680 buff/cache  
KiB Swap: 2017276 total, 2017276 free, 0 used, 579216 avail Mem

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
815	www-data	20	0	826868	5780	3592	S	21.3	0.6	0:16.72	apache2
816	www-data	20	0	826812	5732	3592	S	21.3	0.6	0:16.86	apache2
7	root	20	0	0	0	0	R	1.3	0.0	0:01.63	ksoftirqd/0
8	root	20	0	0	0	0	I	0.3	0.0	0:00.91	rcu_sched
1498	joselep+	20	0	42672	3988	3408	R	0.3	0.4	0:11.37	top
1	root	20	0	77936	8856	6504	S	0.0	0.9	0:01.90	systemd
2	root	20	0	0	0	0	S	0.0	0.0	0:00.00	kthreadd
4	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworke/0:0H
6	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	mm_percpu_wq
9	root	20	0	0	0	0	I	0.0	0.0	0:00.00	rcu_bh

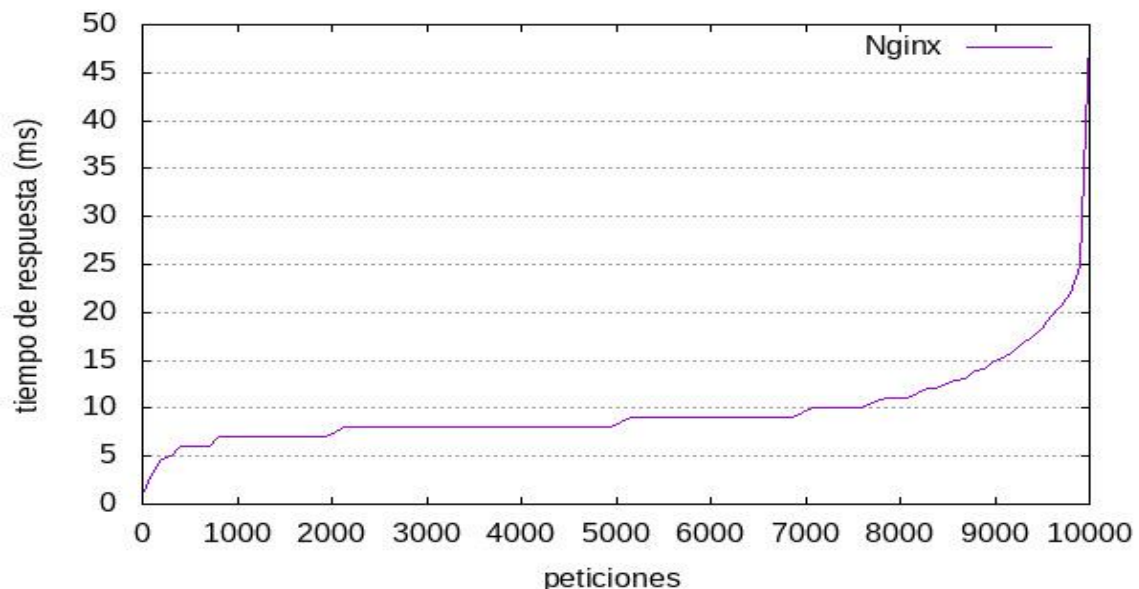
m3 [Corriendo] - Oracle VM VirtualBox

Archivo Máquina Ver Entrada Dispositivos Ayuda

top - 22:22:15 up 2:23, 1 user, load average: 0.15, 0.07, 0.04  
Tasks: 88 total, 3 running, 46 sleeping, 0 stopped, 0 zombie  
%Cpu(s): 16.5 us, 40.5 sy, 0.0 ni, 1.0 id, 0.0 wa, 0.0 hi, 41.9 si, 0.0 st  
KiB Mem : 1008804 total, 303408 free, 121508 used, 583888 buff/cache  
KiB Swap: 2017276 total, 2017276 free, 0 used, 741908 avail Mem

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
1827	www-data	20	0	143848	7104	5184	R	96.0	0.7	0:07.16	nginx
7	root	20	0	0	0	0	S	2.0	0.0	0:02.55	ksoftirqd/0
8	root	20	0	0	0	0	R	0.3	0.0	0:01.04	rcu_sched
1723	root	20	0	0	0	0	I	0.3	0.0	0:00.21	kworke/0:0
1	root	20	0	159912	9220	6752	S	0.0	0.9	0:03.22	systemd
2	root	20	0	0	0	0	S	0.0	0.0	0:00.00	kthreadd
4	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworke/0:0H
6	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	mm_percpu_wq
9	root	20	0	0	0	0	I	0.0	0.0	0:00.00	rcu_bh

10000 peticiones, 10 peticiones concurrentes



Servidores Web de Altas Prestaciones

```

Benchmarking 192.168.56.103 (be patient)
Completed 1000 requests
Completed 2000 requests
Completed 3000 requests
Completed 4000 requests
Completed 5000 requests
Completed 6000 requests
Completed 7000 requests
Completed 8000 requests
Completed 9000 requests
Completed 10000 requests
Finished 10000 requests


Server Software:      nginx/1.14.0
Server Hostname:      192.168.56.103
Server Port:          80


Document Path:        /ejemplo.html
Document Length:      79 bytes


Concurrency Level:    10
Time taken for tests:  9.598 seconds
Complete requests:    10000
Failed requests:      0
Total transferred:    3470000 bytes
HTML transferred:     790000 bytes
Requests per second:  1041.89 [#/sec] (mean)
Time per request:     9.598 [ms] (mean)
Time per request:     0.960 [ms] (mean, across all concurrent requests)
Transfer rate:        353.06 [Kbytes/sec] received


Connection Times (ms)
              min    mean[+/-sd] median    max
Connect:        0      0    0.8      0     15
Processing:      1      9    4.0      8     48
Waiting:         1      9    4.0      8     47
Total:          1     10    4.1      8     49


Percentage of the requests served within a certain time (ms)
 50%      8
 66%      9
 75%     10
 80%     11
 90%     15
 95%     18
 98%     22
 99%     25
100%     49 (longest request)

```

## - HAProxy:

m1 [Corriendo] - Oracle VM VirtualBox

Archivo Máquina Ver Entrada Dispositivos Ayuda

top - 22:25:51 up 2:29, 1 user, load average: 0,00, 0,02, 0,00  
 tasks: 89 total, 1 running, 49 sleeping, 0 stopped, 0 zombie  
 %cpu(s): 10,7 us, 19,3 sy, 0,0 ni, 53,7 id, 0,0 wa, 0,0 hi, 16,3 si, 0,0 st  
 KiB Mem : 1008804 total, 167008 free, 284664 used, 557182 buff/cache  
 KiB Swap: 2017276 total, 2017276 free, 0 used, 579204 avail Mem

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
893	www-data	20	0	826812	5668	3532	S	19,9	0,6	0:16,80	apache2
894	www-data	20	0	826812	5668	3536	S	18,6	0,6	0:16,69	apache2
7	root	20	0	0	0	0	S	1,0	0,0	0:01,59	ksoftirqd/0
1606	Joselep+	20	0	42672	3780	3200	R	0,3	0,4	0:11,26	top
1	root	20	0	77940	8916	6556	S	0,0	0,9	0:01,88	systemd
2	root	20	0	0	0	0	S	0,0	0,0	0:00,58	kthreadd
4	root	0	-20	0	0	0	I	0,0	0,0	0:00,00	kuworker/0:0H
6	root	0	-20	0	0	0	I	0,0	0,0	0:00,00	mm_percpu_wq
8	root	20	0	0	0	0	I	0,0	0,0	0:00,86	rcu_sched

m2 [Corriendo] - Oracle VM VirtualBox

Archivo Máquina Ver Entrada Dispositivos Ayuda

top - 22:25:49 up 2:29, 1 user, load average: 0,05, 0,02, 0,00  
 tasks: 89 total, 3 running, 48 sleeping, 0 stopped, 0 zombie  
 %cpu(s): 13,1 us, 18,5 sy, 0,0 ni, 49,1 id, 0,0 wa, 0,0 hi, 19,3 si, 0,0 st  
 KiB Mem : 1008804 total, 157088 free, 284636 used, 567080 buff/cache  
 KiB Swap: 2017276 total, 2017276 free, 0 used, 579204 avail Mem

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
816	www-data	20	0	826812	5732	3592	S	20,9	0,6	0:18,11	apache2
815	www-data	20	0	826868	5780	3592	S	19,9	0,6	0:17,96	apache2
7	root	20	0	0	0	0	R	1,3	0,0	0:01,72	ksoftirqd/0
32	root	20	0	0	0	0	I	0,3	0,0	0:04,24	kuworker/0:1
1498	Joselep+	20	0	42672	3988	3408	R	0,3	0,4	0:11,69	top
1	root	20	0	77936	8856	6504	S	0,0	0,9	0:01,50	systemd
2	root	20	0	0	0	0	S	0,0	0,0	0:00,00	kthreadd
4	root	0	-20	0	0	0	I	0,0	0,0	0:00,00	kuworker/0:0H
6	root	0	-20	0	0	0	I	0,0	0,0	0:00,00	mm_percpu_wq
8	root	20	0	0	0	0	R	0,0	0,0	0:00,93	rcu_sched

m3 [Corriendo] - Oracle VM VirtualBox

Archivo Máquina Ver Entrada Dispositivos Ayuda

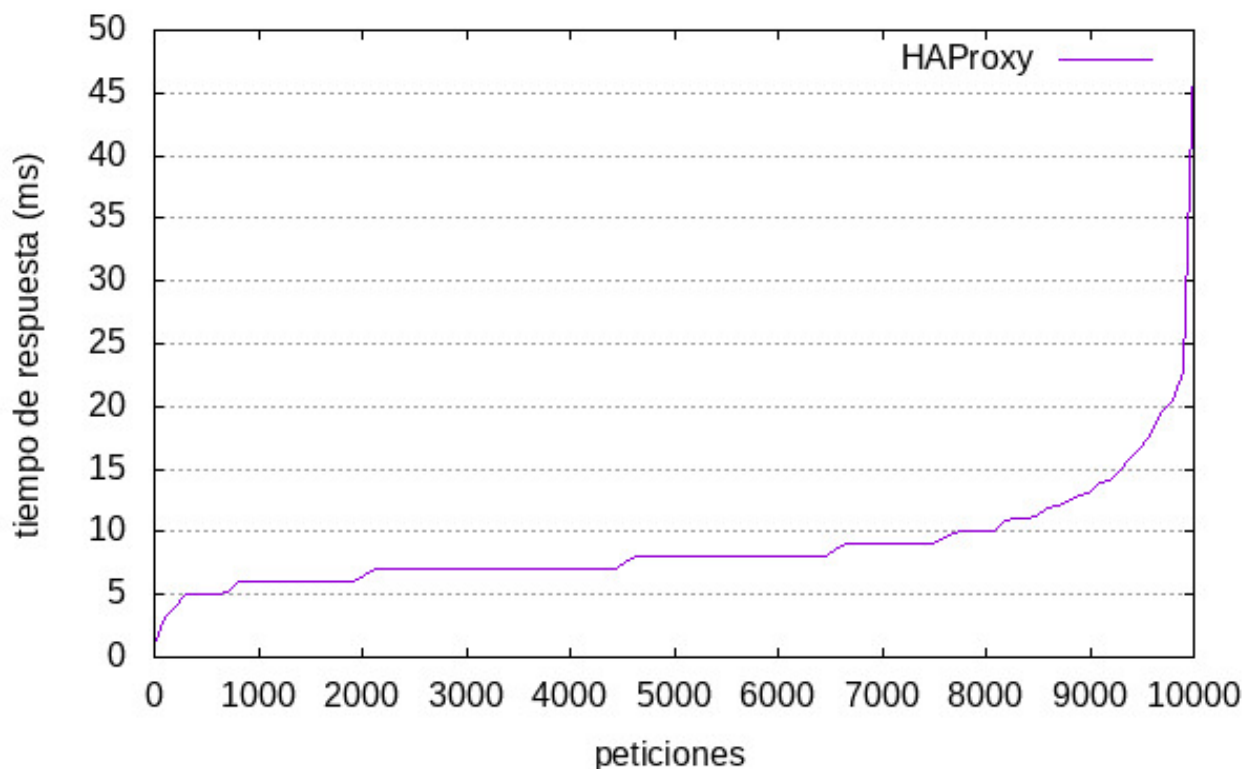
top - 22:25:52 up 2:27, 1 user, load average: 0,08, 0,04, 0,03  
 tasks: 88 total, 4 running, 45 sleeping, 0 stopped, 0 zombie  
 %cpu(s): 21,0 us, 41,2 sy, 0,0 ni, 0,7 id, 0,0 wa, 0,0 hi, 37,1 si, 0,0 st  
 KiB Mem : 1008804 total, 301376 free, 121632 used, 585796 buff/cache  
 KiB Swap: 2017276 total, 2017276 free, 0 used, 741684 avail Mem

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
1938	haproxy	20	0	54524	2572	1240	R	74,3	0,3	0:05,03	haproxy
1770	syslog	20	0	263036	4380	3600	S	22,0	0,4	0:05,12	rsyslogd
7	root	20	0	0	0	0	R	2,0	0,0	0:02,73	ksoftirqd/0
8	root	20	0	0	0	0	R	0,3	0,0	0:01,10	rcu_sched
1723	root	20	0	0	0	0	I	0,3	0,0	0:00,34	kuworker/0:0
1	root	20	0	159912	9220	6752	S	0,0	0,9	0:04,25	systemd
2	root	20	0	0	0	0	S	0,0	0,0	0:00,00	kthreadd
4	root	0	-20	0	0	0	I	0,0	0,0	0:00,00	kuworker/0:0H
6	root	0	-20	0	0	0	I	0,0	0,0	0:00,00	mm_percpu_wq

Símbolo del sistema - ab - g resHaproxy.csv -n 10000 -c 10 http://192.168.56.103/ejemplo.html

C:\wamp64\bin\apache\apache2.4.37\bin>ab -g resHaproxy.csv -n 10000 -c 10 http://192.168.56.103/ejemplo.htm  
 1  
 This is ApacheBench, Version 2.3 <\$Revision: 1843412 \$>  
 Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/  
 Licensed to The Apache Software Foundation, http://www.apache.org/  
  
 Benchmarking 192.168.56.103 (be patient)  
 Completed 1000 requests  
 Completed 2000 requests  
 Completed 3000 requests  
 Completed 4000 requests  
 Completed 5000 requests  
 Completed 6000 requests  
 Completed 7000 requests  
 Completed 8000 requests

10000 peticiones, 10 peticiones concurrentes



Servidores Web de Altas Prestaciones

Práctica 3

Realizado por: José Luis Pedraza Roman

```

Benchmarking 192.168.56.103 (be patient)
Completed 1000 requests
Completed 2000 requests
Completed 3000 requests
Completed 4000 requests
Completed 5000 requests
Completed 6000 requests
Completed 7000 requests
Completed 8000 requests
Completed 9000 requests
Completed 10000 requests
Finished 10000 requests


Server Software:      Apache/2.4.29
Server Hostname:      192.168.56.103
Server Port:          80


Document Path:        /ejemplo.html
Document Length:      79 bytes


Concurrency Level:     10
Time taken for tests:  8.662 seconds
Complete requests:     10000
Failed requests:       0
Total transferred:     3480000 bytes
HTML transferred:      790000 bytes
Requests per second:   1154.47 [#/sec] (mean)
Time per request:      8.662 [ms] (mean)
Time per request:      0.866 [ms] (mean, across all concurrent requests)
Transfer rate:         392.34 [Kbytes/sec] received


Connection Times (ms)
              min    mean[+/-sd] median    max
Connect:        0      0    0.8      0     15
Processing:      1      8    3.7      7     48
Waiting:         1      8    3.6      7     47
Total:           1      9    3.8      8     48


Percentage of the requests served within a certain time (ms)
 50%      8
 66%      9
 75%      9
 80%     10
 90%     13
 95%     17
 98%     21
 99%     23
100%     48 (longest request)

```



## - Pound:

m1 [Corriendo] - Oracle VM VirtualBox

Archivo Máquina Ver Entrada Dispositivos Ayuda

```
top - 22:13:34 up 2:17, 1 user, load average: 0,00, 0,00, 0,00
Tasks: 89 total, 2 running, 49 sleeping, 0 stopped, 0 zombie
%Cpu(s): 5,3 us, 8,7 sy, 0,0 ni, 75,1 id, 0,0 wa, 0,0 hi, 10,9 si, 0,0 st
KiB Mem : 1008804 total, 167636 free, 285180 used, 555388 buff/cache
KiB Swap: 2017276 total, 2017276 free, 0 used, 578704 avail Mem
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
893	www-data	20	0	826756	5612	3532	S	11,3	0,6	0:13.51	apache2
894	www-data	20	0	826812	5660	3536	S	11,0	0,6	0:13.44	apache2
7	root	20	0	0	0	0	S	1,0	0,0	0:01.36	ksoftirqd/0
8	root	20	0	0	0	0	R	0,3	0,0	0:00.81	rcu_sched
1606	joselep+	20	0	42672	3780	3200	R	0,3	0,4	0:10.22	top
1	root	20	0	77940	8916	6556	S	0,0	0,9	0:01.88	systemd
2	root	20	0	0	0	0	S	0,0	0,0	0:00.58	kthreadd
4	root	0	-20	0	0	0	I	0,0	0,0	0:00.00	kworker/0:0H
6	root	0	-20	0	0	0	I	0,0	0,0	0:00.00	mm_percpu_wq

m2 [Corriendo] - Oracle VM VirtualBox

Archivo Máquina Ver Entrada Dispositivos Ayuda

```
top - 22:13:32 up 2:16, 1 user, load average: 0,33, 0,09, 0,03
Tasks: 89 total, 2 running, 49 sleeping, 0 stopped, 0 zombie
%Cpu(s): 6,0 us, 10,7 sy, 0,0 ni, 68,3 id, 0,0 wa, 0,0 hi, 14,9 si, 0,0 st
KiB Mem : 1008804 total, 157840 free, 285008 used, 565956 buff/cache
KiB Swap: 2017276 total, 2017276 free, 0 used, 578884 avail Mem
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
815	www-data	20	0	826812	5736	3592	S	12,6	0,6	0:14.52	apache2
816	www-data	20	0	826812	5732	3592	S	11,6	0,6	0:14.61	apache2
7	root	20	0	0	0	0	S	1,0	0,0	0:01.47	ksoftirqd/0
8	root	20	0	0	0	0	R	0,3	0,0	0:00.87	rcu_sched
1	root	20	0	77936	8856	6504	S	0,0	0,9	0:01.90	systemd
2	root	20	0	0	0	0	S	0,0	0,0	0:00.00	kthreadd
4	root	0	-20	0	0	0	I	0,0	0,0	0:00.00	kworker/0:0H
6	root	0	-20	0	0	0	I	0,0	0,0	0:00.00	mm_percpu_wq
9	root	20	0	0	0	0	I	0,0	0,0	0:00.00	rcu_bh
10	root	rt	0	0	0	0	S	0,0	0,0	0:00.00	migration/0

m3 [Corriendo] - Oracle VM VirtualBox

Archivo Máquina Ver Entrada Dispositivos Ayuda

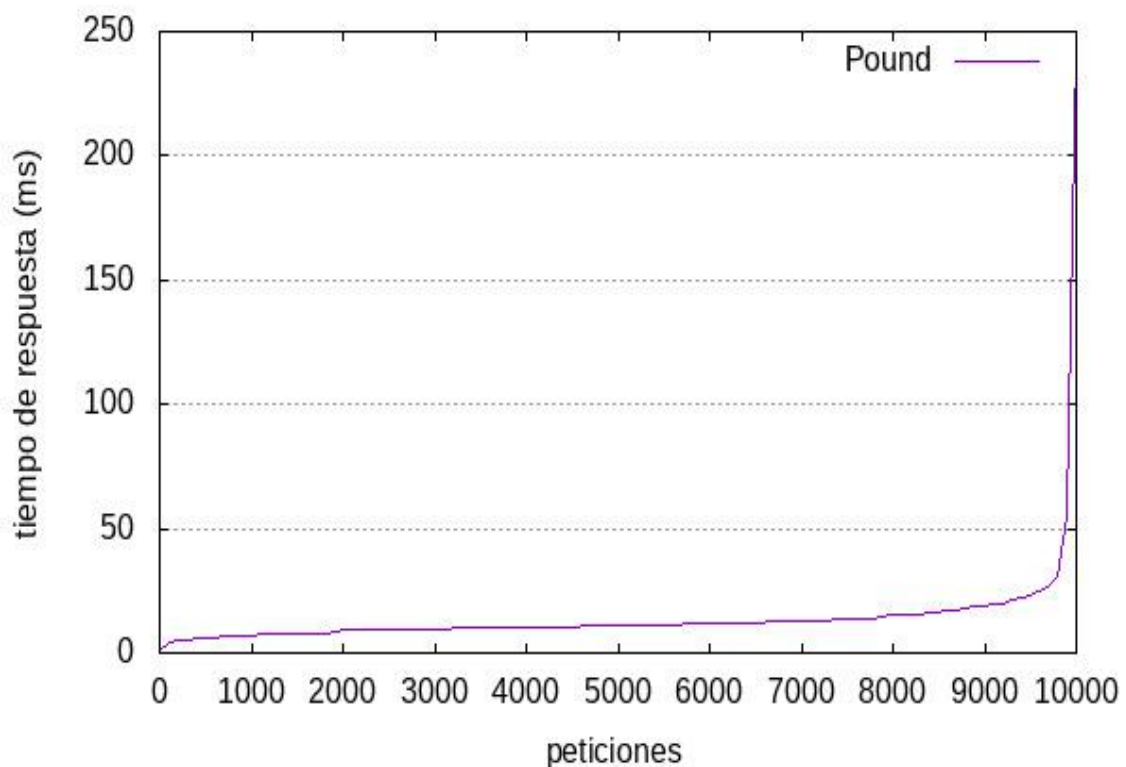
```
top - 22:13:33 up 2:14, 1 user, load average: 0,65, 0,23, 0,08
Tasks: 87 total, 4 running, 45 sleeping, 0 stopped, 0 zombie
%Cpu(s): 26,1 us, 36,9 sy, 0,0 ni, 0,3 id, 0,0 wa, 0,0 hi, 36,6 si, 0,0 st
KiB Mem : 1008804 total, 290660 free, 135476 used, 582668 buff/cache
KiB Swap: 2017276 total, 2017276 free, 0 used, 727992 avail Mem
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
1518	www-data	20	0	591024	3540	1292	R	78,7	0,4	1:04.00	pound
402	root	19	-1	133060	45180	44472	S	13,6	4,5	0:14.40	systemd-journal
770	syslog	20	0	263036	4300	3600	S	4,3	0,4	0:00.49	rsyslogd
7	root	20	0	0	0	0	R	1,3	0,0	0:02.32	ksoftirqd/0
1	root	20	0	159892	9208	6752	S	0,0	0,9	0:02.65	systemd
2	root	20	0	0	0	0	S	0,0	0,0	0:00.00	kthreadd
4	root	0	-20	0	0	0	I	0,0	0,0	0:00.00	kworker/0:0H
6	root	0	-20	0	0	0	I	0,0	0,0	0:00.00	mm_percpu_wq
8	root	20	0	0	0	0	R	0,0	0,0	0:00.94	rcu_sched

```
C:\wamp64\bin\apache\apache2.4.37\bin>ab -g resPound.csv -n 10000 -c 10 http://192.168.56.103/ejemplo.html
This is ApacheBench, Version 2.3 <$Revision: 1843412 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking 192.168.56.103 (be patient)
Completed 1000 requests
Completed 2000 requests
Completed 3000 requests
Completed 4000 requests
Completed 5000 requests
Completed 6000 requests
Completed 7000 requests
Completed 8000 requests
```

10000 peticiones, 10 peticiones concurrentes



Servidores Web de Altas Prestaciones

Práctica 3

Realizado por: José Luis Pedraza Roman

```

Benchmarking 192.168.56.103 (be patient)
Completed 1000 requests
Completed 2000 requests
Completed 3000 requests
Completed 4000 requests
Completed 5000 requests
Completed 6000 requests
Completed 7000 requests
Completed 8000 requests
Completed 9000 requests
Completed 10000 requests
Finished 10000 requests


Server Software:      Apache/2.4.29
Server Hostname:      192.168.56.103
Server Port:          80


Document Path:        /ejemplo.html
Document Length:      79 bytes


Concurrency Level:     10
Time taken for tests:  12.874 seconds
Complete requests:     10000
Failed requests:       0
Total transferred:     3480000 bytes
HTML transferred:     790000 bytes
Requests per second:   776.74 [#/sec] (mean)
Time per request:      12.874 [ms] (mean)
Time per request:      1.287 [ms] (mean, across all concurrent requests)
Transfer rate:         263.97 [Kbytes/sec] received


Connection Times (ms)
              min    mean[+/-sd] median    max
Connect:        0      0    0.7      0     16
Processing:     1     12   11.6     11    247
Waiting:        1     10    9.2      9    235
Total:          1     13   11.6     11    247


Percentage of the requests served within a certain time (ms)
 50%      11
 66%      12
 75%      14
 80%      15
 90%      19
 95%      23
 98%      31
 99%      55
100%     247 (longest request)

```

## 7) Conclusión

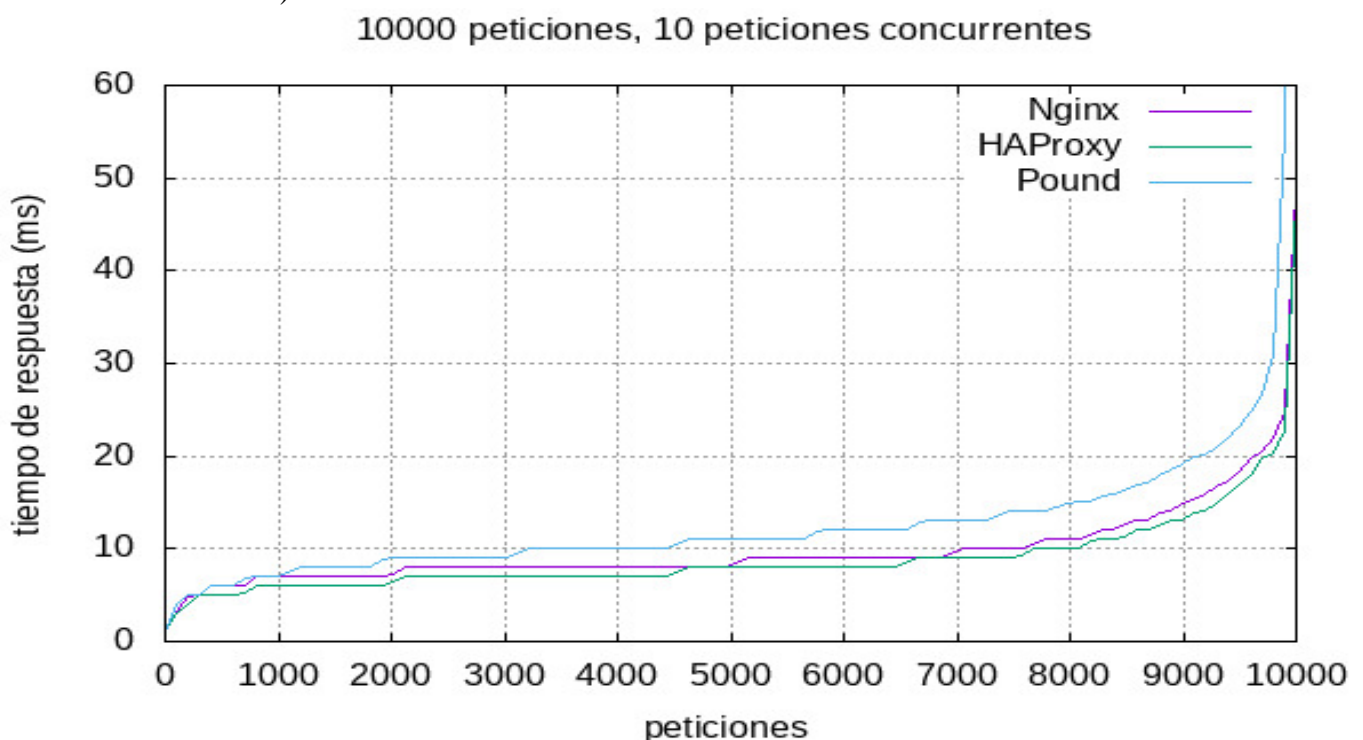
Con el siguiente script de gnuplot grafico las salidas .csv devueltas anteriormente para comparar gráficamente los resultados de las diferentes herramientas.

```
1 set terminal png size 600
2 set output "resultadosAll.png"
3 set title "10000 peticiones, 10 peticiones concurrentes"
4 set size ratio 0.6
5 set grid y
6 set grid x
7 set yrange [0:60]
8 set xlabel "peticiones"
9 set ylabel "tiempo de respuesta (ms)"
10 plot "resNginx.csv" using 9 smooth sbezier with lines title "Nginx",
11 "resHaproxy.csv" using 9 smooth sbezier with lines title "HAProxy",
12 "resPound.csv" using 9 smooth sbezier with lines title "Pound"
```

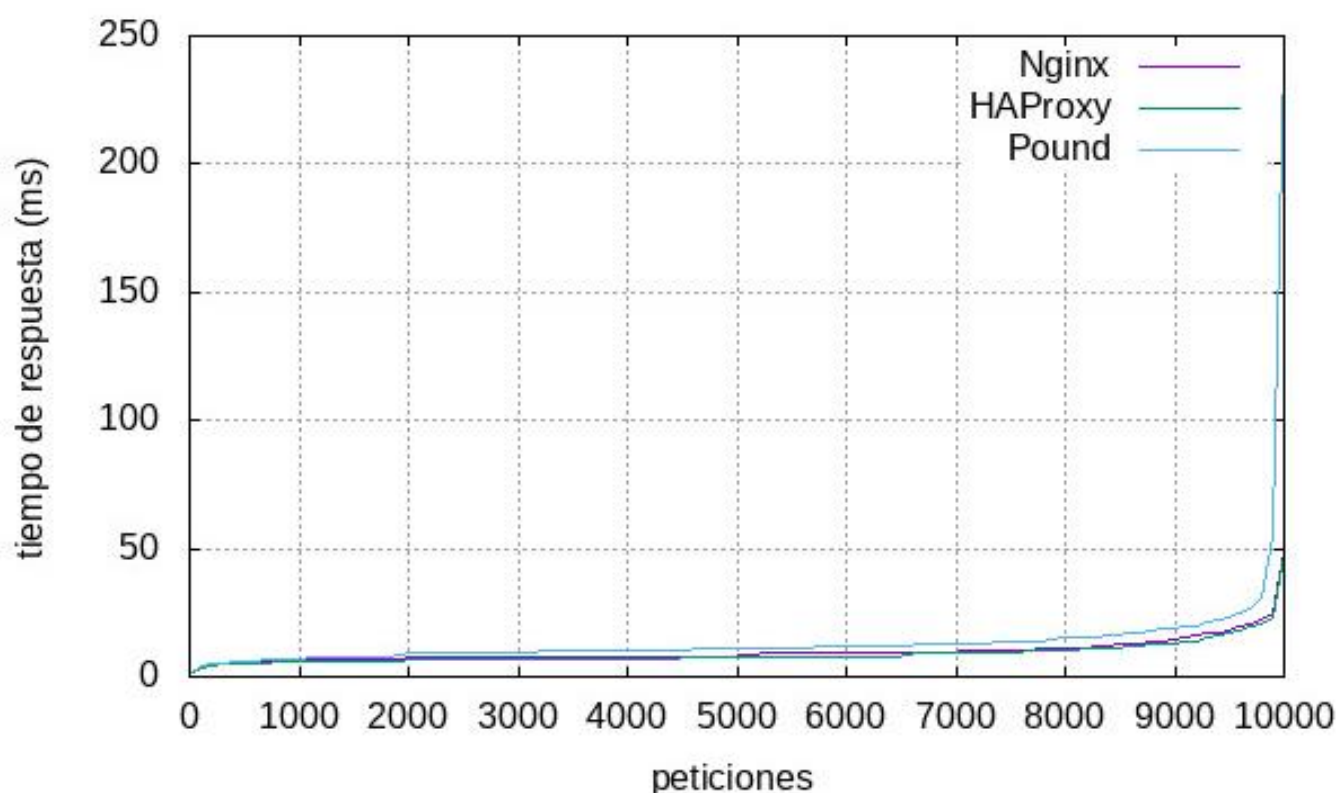
Como podemos ver en las siguientes gráficas comparativas el “mejor” balanceador de carga según las pruebas realizadas es la herramienta HAProxy, justo después Nginx y a una gran distancia de estos dos, Pound (seguramente por el tipo de configuración básica aplicado en este último caso).

Con HAProxy los tiempos de ejecución, por petición, de conexión, procesamiento, espera y total son los menores respecto a las otras dos herramientas probadas (aunque con muy poca diferencia respecto a Nginx, cuyas dos configuraciones están muy parejas en los benchmarks).

Esto lo podemos apreciar claramente en la siguiente gráfica ampliada (ya que no llegamos a ver el tiempo de respuesta ante el máximo número de peticiones de Pound).



### 10000 peticiones, 10 peticiones concurrentes



En último lugar tenemos Pound, con resultados poco satisfactorios, llegando hasta 247ms cuando se le mandan el 100% de peticiones concurrentes, siendo este valor mucho menor para Nginx (49ms longest request) y HAProxy (48ms longest request). Como vemos en las gráficas es el peor con diferencia, HAProxy y Nginx distribuyen la misma carga de trabajo mucho más rápido (quizás sea por lo que son los más usados para esta tarea).

Balancedador de carga	Número de peticiones	Duración del test	Peticiones/s	Tiempo por peticiones concurrentes
HAProxy	10000	8.662 sec	392.34 Kbytes/sec	0.866 ms
Nginx	10000	9.598 sec	353.06 Kbytes/sec	0.960 ms
Pound	10000	12.874 sec	263.97 Kbytes/sec	1.287 ms