



# SERVIDORES WEB DE ALTAS PRESTACIONES

---

## PRÁCTICA 3: Balanceo de carga en un sitio web

**Autor**

Miguel Ángel Pérez Díaz



Escuela Técnica Superior de Ingenierías Informática y  
de Telecomunicación

—  
Granada, 2020

## 1. Configurar una máquina e instalar el nginx como balanceador de carga.

Siguiendo el esquema de la práctica propuesta utilizaremos M1 y M2 como granja web, una nueva máquina M3 como balanceador y M4 como máquina que realiza peticiones al balanceador. Inicialmente vamos a definir los elementos utilizados en esta práctica:

- Una nueva máquina virtual M3 con toda la configuración de red ya configurada.

```
Configuración de perfil [ Help ]

Enter the username and password you will use to log in to the system. You can
configure SSH access on the next screen but a password is still needed for
sudo.

Your name: MIGUEL ANGEL PREZ DIAZ

Your server's name: m3
The name it uses when it talks to other computers.

Pick a username: miguel444

Choose a password: *****

Confirm your password: *****

[ Hecho ]
```

- En mi caso en lugar de crear otra máquina M4 para realizar las peticiones, voy a trabajar con la máquina anfitrión utilizando la bash de Ubuntu desde Windows.
- Las máquinas virtuales M1 y M2 previamente creadas y configuradas en las prácticas anteriores.

Una vez definidos los principales elementos de nuestra práctica, vamos a tratar de utilizar M3 como balanceador de carga de nuestra granja web, para ello debemos inicialmente instalar el servidor web **nginx**:

- *sudo apt-get update && sudo apt-get dist-upgrade && sudo apt-get autoremove*

```
miguel444@m3:~$ sudo apt-get update && sudo apt-get dist-upgrade && sudo apt-get autoremove
```

- *sudo apt-get install nginx*

```
miguel444@m3:~$ sudo apt-get install nginx
```

- *sudo systemctl start nginx*

```
miguel444@m3:~$ sudo systemctl start nginx
```

Una vez instalado nginx y activado el servicio, vamos a tratar de configurarlo para poder llevar a cabo el balanceo de la carga. En nuestro caso la configuración la vamos a hacer para definir balanceo por **round robin** (asignación por turnos) y por **ponderaciones o pesos** de los servidores.

Para configurar nuestro balanceador debemos modificar, o en nuestro caso ha sido necesario crear, el fichero */etc/nginx/conf.d/default.conf*.

En el fichero indicaremos inicialmente que servidores formarán nuestra granja web, para ello indicaremos sus correspondientes IPs en la sección ‘upstream’ del fichero. Después en la sección ‘server’ indicaremos el puerto de escucha y el nombre del servidor, así como que la conexión entre nginx y los servidores finales sea HTTP 1.1 así como especificarle que debe eliminar la cabecera Connection (hacerla vacía) para evitar que se pase al servidor final la cabecera que indica el usuario.

Finalmente, el fichero quedaría de la siguiente forma:

```
GNU nano 2.9.3 /etc/nginx/conf.d/default.conf

upstream servidoresSWAP{
    server 192.168.56.12;
    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://servidoresSWAP;
        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 "";
    }
}

[ Read 24 lines ]
^G Get Help  ^O Write Out  ^W Where Is   ^K Cut Text   ^J Justify    ^C Cur Pos    M-U Undo
^X Exit      ^R Read File  ^N Replace    ^U Uncut Text ^T To Spell   ^G Go To Line M-E Redo
```

En la configuración anterior hemos definido el balanceo por round robin en la que todos los servidores de la granja tienen la misma prioridad. Una vez configurado todo vamos a tratar de probar nuestro balanceador de carga, para ello reiniciaremos el servicio con: *sudo service nginx restart* (comando que utilizaremos para reiniciar el servicio cada vez que hagamos un cambio en el archivo de configuración).

Una vez reiniciado el servicio, vamos a tratar de usar el comando cURL en nuestra máquina anfitriona M4 indicando la dirección IP de balanceador:

```
mapd0004@LAPTOP-9FUP9E1A:~$ curl http://192.168.56.104
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
  body {
    width: 35em;
    margin: 0 auto;
    font-family: Tahoma, Verdana, Arial, sans-serif;
  }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

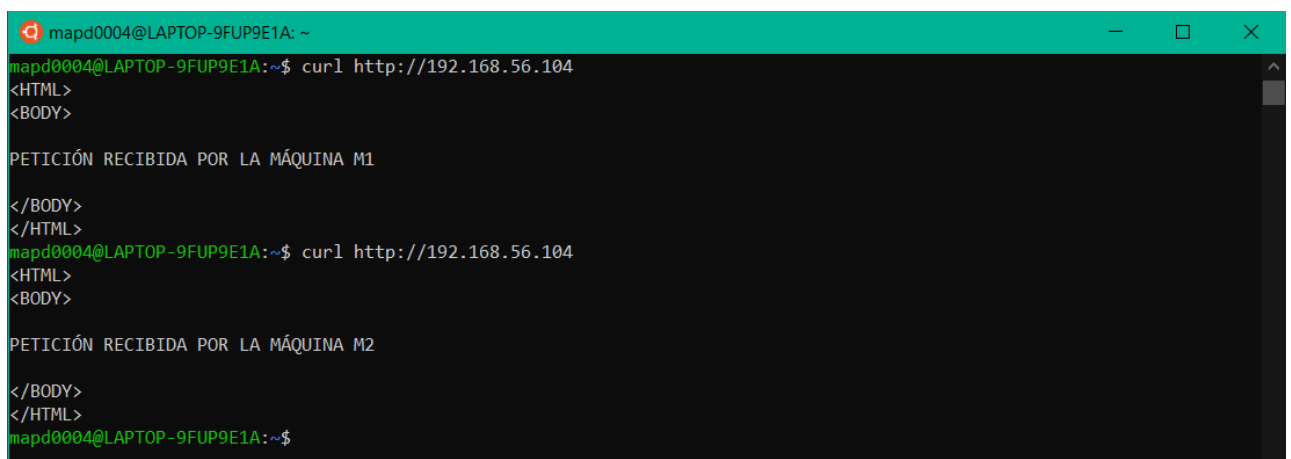
<p><em>Thank you for using nginx.</em></p>
</body>
</html>
```

Podemos observar que no obtenemos una respuesta correcta de nuestro balanceador, esto se debe a que tenemos nginx configurado como servidor web y nos está devolviendo el archivo index.html de nginx y no de las máquinas que forman la granja. Para solucionar esto basta con comentar la línea: `include /etc/nginx/sites-enabled/*` del archivo `/etc/nginx/nginx.conf`.

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



Una vez modificado el fichero, volvemos a reiniciar el servicio y volvemos a probar el comando cURL hacia la máquina balanceadora, y podemos ver como en este caso si se obtiene la respuesta esperada:



```
mapd0004@LAPTOP-9FUP9E1A: ~
mapd0004@LAPTOP-9FUP9E1A:~$ curl http://192.168.56.104
<HTML>
<BODY>

PETICIÓN RECIBIDA POR LA MÁQUINA M1

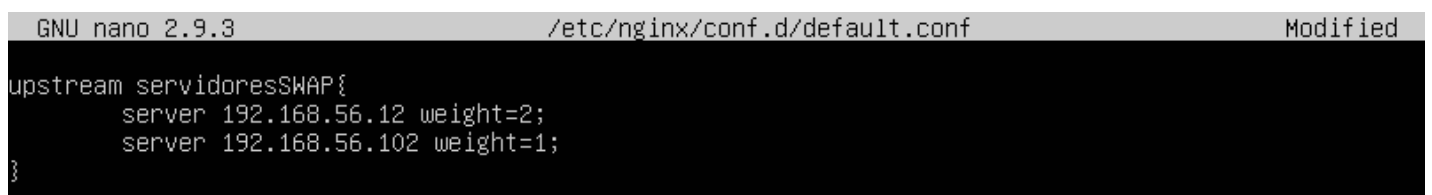
</BODY>
</HTML>
mapd0004@LAPTOP-9FUP9E1A:~$ curl http://192.168.56.104
<HTML>
<BODY>

PETICIÓN RECIBIDA POR LA MÁQUINA M2

</BODY>
</HTML>
mapd0004@LAPTOP-9FUP9E1A:~$
```

Como segunda parte de esta sección debemos realizar un balanceo por ponderaciones o pesos, para ello bastaría con modificar ligeramente el archivo de configuración creado anteriormente, especificando los pesos para cada uno de los servidores finales, dando en este caso el doble de capacidad a la máquina M1 sobre M2:

```
GNU nano 2.9.3 /etc/nginx/conf.d/default.conf Modified
upstream servidoresSWAP{
    server 192.168.56.12 weight=2;
    server 192.168.56.102 weight=1;
}
```



Podemos ver como se han definido los pesos con la variable 'weight'.

Una vez realizado este cambio, volvemos a reiniciar el servicio y realizamos cURL desde la máquina anfitriona al balanceador:

```
mapd0004@LAPTOP-9FUP9E1A: ~  
mapd0004@LAPTOP-9FUP9E1A:~$ curl http://192.168.56.104  
<HTML>  
<BODY>  
  
PETICIÓN RECIBIDA POR LA MÁQUINA M1  
  
</BODY>  
</HTML>  
mapd0004@LAPTOP-9FUP9E1A:~$ curl http://192.168.56.104  
<HTML>  
<BODY>  
  
PETICIÓN RECIBIDA POR LA MÁQUINA M1  
  
</BODY>  
</HTML>  
mapd0004@LAPTOP-9FUP9E1A:~$ curl http://192.168.56.104  
<HTML>  
<BODY>  
  
PETICIÓN RECIBIDA POR LA MÁQUINA M2  
  
</BODY>  
</HTML>  
mapd0004@LAPTOP-9FUP9E1A:~$
```

Podemos observar cómo se le da más prioridad a la máquina M1 recibiendo 2 peticiones de cada 3 enviadas al balanceador.

## 2. ***Configurar una máquina e instalar el haproxy como balanceador de carga.***

En esta nueva sección vamos a realizar las mismas tareas que en la sección anterior. Para ello inicialmente debemos instalar el balanceador haproxy usando el comando:

- *Sudo apt-get install haproxy*

```
miguel1444@m3:~$ sudo apt-get install haproxy_
```

Una vez instalado el balanceador, debemos configurarlo para poder llevar a cabo la tarea. En este caso debemos modificar el fichero `/etc/haproxy/haproxy.cfg` para indicarle cuales son nuestros balanceadores y que peticiones balancear.

Vamos a indicar que las peticiones las escuche por el puerto 80 y que las redirija a la dirección ip de la máquina correspondiente. Finalmente, el fichero de configuración quedaría de la siguiente manera:

```
GNU nano 2.9.3 /etc/haproxy/haproxy.cfg Modified
# 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$
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
    balance roundrobin
    server m1 192.168.56.12:80 maxconn 32
    server m2 192.168.56.102:80 maxconn 32
```

Al mismo tiempo se le ha indicado que realice un balanceo por round robin con la variable ‘balance roundrobin’ en la sección backend.

Antes de probar el funcionamiento del balanceador haproxy, se debe pausar el balanceador nginx instalado anteriormente ya que ambos escuchan por el puerto 80 y los dos no pueden estar funcionando. Y después reiniciaremos el servicio de haproxy para que empiece a funcionar:

```
miguel444@m3:~$ sudo service nginx stop
miguel444@m3:~$ sudo service haproxy restart
miguel444@m3:~$ sudo service haproxy status
• haproxy.service - HAProxy Load Balancer
   Loaded: loaded (/lib/systemd/system/haproxy.service; enabled; vendor preset: enabled)
   Active: active (running) since Thu 2020-04-02 11:42:40 UTC; 1min 48s ago
     Docs: man:haproxy(1)
           file:/usr/share/doc/haproxy/configuration.txt.gz
   Process: 1906 ExecStartPre=/usr/sbin/haproxy -f $CONFIG -c -q $EXTRA_OPTS (code=exited, status=0/SUCCESS)
   Main PID: 1912 (haproxy)
     Tasks: 2 (limit: 503)
    CGroup: /system.slice/haproxy.service
            └─1912 /usr/sbin/haproxy -Ws -f /etc/haproxy/haproxy.cfg -p /run/haproxy.pid
              1917 /usr/sbin/haproxy -Ws -f /etc/haproxy/haproxy.cfg -p /run/haproxy.pid

abr 02 11:42:40 m3 systemd[1]: Stopping HAProxy Load Balancer...
abr 02 11:42:40 m3 systemd[1]: Stopped HAProxy Load Balancer.
abr 02 11:42:40 m3 systemd[1]: Starting HAProxy Load Balancer...
abr 02 11:42:40 m3 haproxy[1912]: Proxy http-in started.
abr 02 11:42:40 m3 haproxy[1912]: Proxy http-in started.
abr 02 11:42:40 m3 haproxy[1912]: Proxy servidoresSWAP started.
abr 02 11:42:40 m3 haproxy[1912]: Proxy servidoresSWAP started.
abr 02 11:42:40 m3 systemd[1]: Started HAProxy Load Balancer.
lines 1-20/20 (END)
```

Todo parece correcto y en funcionamiento por lo que procederemos a realizar las peticiones con el comando cURL desde la máquina anfitriona, obteniéndose así la siguiente respuesta:

```
mapd0004@LAPTOP-9FUP9E1A: ~  
mapd0004@LAPTOP-9FUP9E1A:~$ curl http://192.168.56.104  
<HTML>  
<BODY>  
  
PETICIÓN RECIBIDA POR LA MÁQUINA M1  
  
</BODY>  
</HTML>  
mapd0004@LAPTOP-9FUP9E1A:~$ curl http://192.168.56.104  
<HTML>  
<BODY>  
  
PETICIÓN RECIBIDA POR LA MÁQUINA M2  
  
</BODY>  
</HTML>  
mapd0004@LAPTOP-9FUP9E1A:~$ curl http://192.168.56.104  
<HTML>  
<BODY>  
  
PETICIÓN RECIBIDA POR LA MÁQUINA M1  
  
</BODY>  
</HTML>  
mapd0004@LAPTOP-9FUP9E1A:~$ curl http://192.168.56.104  
<HTML>  
<BODY>  
  
PETICIÓN RECIBIDA POR LA MÁQUINA M2  
  
</BODY>  
</HTML>  
mapd0004@LAPTOP-9FUP9E1A:~$
```

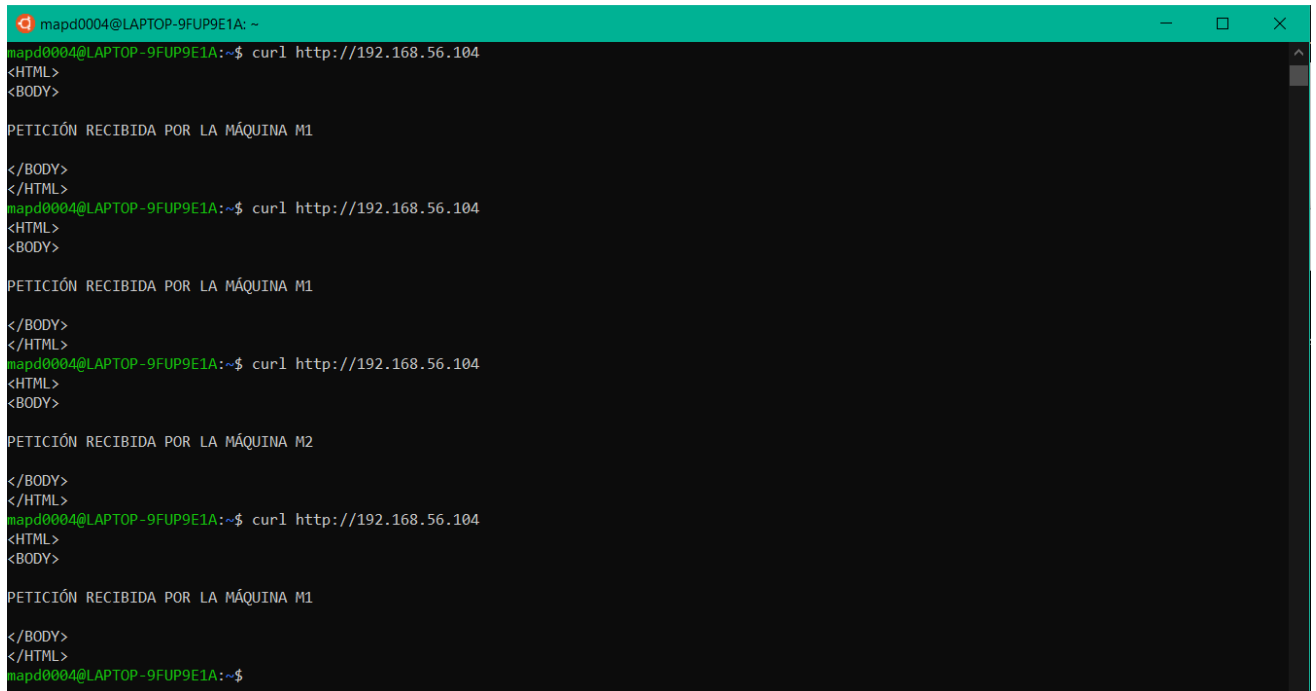
Tras esto, vamos a configurar el balanceador para que realice el balanceo por ponderaciones o pesos, y al igual que antes bastará con una ligera modificación en el archivo de configuración indicando que la máquina M1 tenga el doble de capacidad que M2:

```
GNU nano 2.9.3 /etc/haproxy/haproxy.cfg Modified  
  
# 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$  
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 192.168.56.12:80 maxconn 32 weight 2  
    server m2 192.168.56.102:80 maxconn 32 weight 1  
  
Get Help  Write Out  Where Is  Cut Text  Justify  Cur Pos  M-U Undo  
Exit      Read File  Replace  Uncut Text  To Spell  Go To Line  M-E Redo
```



De la misma manera se ha utilizado la variable ‘weight’ para establecer los pesos de los diferentes servidores finales. Tras esto nuevamente debemos reiniciar el servicio.

Ahora veremos cómo al realizar varias peticiones el balanceador le da más prioridad a la máquina M1:

A terminal window with a teal title bar showing a series of curl requests and responses. The terminal output shows that the first three requests are received by machine M1, and the fourth request is received by machine M2. The responses are in HTML format with a body containing the machine identifier.

```
mapd0004@LAPTOP-9FUP9E1A: ~  
mapd0004@LAPTOP-9FUP9E1A:~$ curl http://192.168.56.104  
<HTML>  
<BODY>  
  
PETICIÓN RECIBIDA POR LA MÁQUINA M1  
  
</BODY>  
</HTML>  
mapd0004@LAPTOP-9FUP9E1A:~$ curl http://192.168.56.104  
<HTML>  
<BODY>  
  
PETICIÓN RECIBIDA POR LA MÁQUINA M1  
  
</BODY>  
</HTML>  
mapd0004@LAPTOP-9FUP9E1A:~$ curl http://192.168.56.104  
<HTML>  
<BODY>  
  
PETICIÓN RECIBIDA POR LA MÁQUINA M1  
  
</BODY>  
</HTML>  
mapd0004@LAPTOP-9FUP9E1A:~$ curl http://192.168.56.104  
<HTML>  
<BODY>  
  
PETICIÓN RECIBIDA POR LA MÁQUINA M2  
  
</BODY>  
</HTML>  
mapd0004@LAPTOP-9FUP9E1A:~$
```

Una vez instalados, configurados y probados tanto *nginx* como *haproxy*, vamos a proceder a realizar una comparación entre tiempos de servicio utilizando ambos utilizando la herramienta Apache Benchmark para simulaciones múltiples peticiones a nuestro balanceador.

### 3. Someter a la granja web a una alta carga, generada con la herramienta Apache Benchmark, teniendo primero nginx y después haproxy.

Inicialmente en mi máquina anfitriona se ha tenido que instalar la herramienta, para ello se ha utilizado el comando: `sudo apt-get install apache2-utils`.

Una vez instalado vamos a realizar una simulación de prueba sobre el balanceador:

`ab -n 10000 -c 10 http://192.168.56.104/index.html`

Realizando 10000 peticiones de manera concurrente con 10 hebras, por lo que se harán de 10 en 10 sobre la ip de la máquina balanceadora.

Para la comparación se ha utilizado la técnica de round robin para ambos balanceadores, obtenido así los siguientes resultados:

Para 10000 peticiones:

TIPO	TIME(s)	REQ/S	TIME/REQ(ms)	TRANSFER RATE(KB/s)	CONNECT(ms)	PROCESSING(ms)	WAITING(ms)	TOTAL(ms)
NGINX	9.868	1013.38	9.868	334.49	0	9	9	10
HAPROXY	10.616	941.97	10.616	311.84	1	10	10	10

Para 100000 peticiones:

TIPO	TIME(s)	REQ/S	TIME/REQ(ms)	TRANSFER RATE(KB/s)	CONNECT(ms)	PROCESSING(ms)	WAITING(ms)	TOTAL(ms)
NGINX	130.410	766.81	13.041	253.11	1	12	12	13
HAPROXY	110.899	901.72	11.090	298.52	1	10	10	11

Se puede observar como cuando el número de peticiones aumenta considerablemente se obtiene una respuesta más rápida por parte del balanceador *haproxy* como así indican los tiempos obtenidos.

Capturas obtenidas con la ejecución para cada balanceador:

```
mapd0004@LAPTOP-9FUP9E1A: ~  
Server Software:      nginx/1.14.0  
Server Hostname:      192.168.56.104  
Server Port:          80  
  
Document Path:        /index.html  
Document Length:      70 bytes  
  
Concurrency Level:    10  
Time taken for tests:  130.410 seconds  
Complete requests:    100000  
Failed requests:      0  
Total transferred:    33800000 bytes  
HTML transferred:    7000000 bytes  
Requests per second:  766.81 [#/sec] (mean)  
Time per request:     13.041 [ms] (mean)  
Time per request:     1.304 [ms] (mean, across all concurrent requests)  
Transfer rate:        253.11 [Kbytes/sec] received  
  
Connection Times (ms)  
  min  mean[+/-sd] median  max  
Connect:    0   1  0.8    0   89  
Processing:  1  12  9.7    8  467  
Waiting:    1  12  9.5    8  466  
Total:      1  13  9.9    8  467  
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%    8  
 66%   15  
 75%   18  
 80%   21  
 90%   25  
 95%   30  
 98%   36  
 99%   40  
100%  467 (longest request)  
mapd0004@LAPTOP-9FUP9E1A:~$
```

**NGINX**

**HAPROXY**

```
mapd0004@LAPTOP-9FUP9E1A: ~  
Server Software:      Apache/2.4.29  
Server Hostname:      192.168.56.104  
Server Port:          80  
  
Document Path:        /index.html  
Document Length:      70 bytes  
  
Concurrency Level:    10  
Time taken for tests:  110.899 seconds  
Complete requests:    100000  
Failed requests:      0  
Total transferred:    33900000 bytes  
HTML transferred:    7000000 bytes  
Requests per second:  901.72 [#/sec] (mean)  
Time per request:     11.090 [ms] (mean)  
Time per request:     1.109 [ms] (mean, across all concurrent requests)  
Transfer rate:        298.52 [Kbytes/sec] received  
  
Connection Times (ms)  
  min  mean[+/-sd] median  max  
Connect:    0   1  0.9    1   17  
Processing:  1  10  7.0    8   63  
Waiting:    1  10  6.9    8   63  
Total:      1  11  7.3    8   64  
  
Percentage of the requests served within a certain time (ms)  
 50%    8  
 66%   12  
 75%   14  
 80%   17  
 90%   21  
 95%   25  
 98%   31  
 99%   35  
100%   64 (longest request)  
mapd0004@LAPTOP-9FUP9E1A:~$
```

- **Se propone el uso de algún otro software de balanceo diferente a los dos explicados en este guion (por ejemplo, Pound)**

En este caso no se ha podido instalar el balanceador **Pound** de la misma forma que los demás con un simple *apt-get install*, sino que ha sido necesario descargar e instalar el balanceador a mano. Para la instalación y configuración se han seguido los siguientes pasos:

- Se ha descargado el fichero .deb correspondiente al balanceador con el comando: *wget*

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

```
root@m3:/home/miguel444# wget http://launchpadlibrarian.net/317629201/pound_2.7-1.3_amd64.deb
--2020-04-04 21:53:00-- http://launchpadlibrarian.net/317629201/pound_2.7-1.3_amd64.deb
Resolving launchpadlibrarian.net (launchpadlibrarian.net)... 91.189.89.229, 91.189.89.228, 2001:67c:
1560:8003::8008, ...
Connecting to launchpadlibrarian.net (launchpadlibrarian.net)|91.189.89.229|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 95406 (93K) [application/x-debian-package]
Saving to: 'pound_2.7-1.3_amd64.deb'

pound_2.7-1.3_amd64.deb 100%[=====] 93,17K 142KB/s in 0,7s
2020-04-04 21:53:02 (142 KB/s) - 'pound_2.7-1.3_amd64.deb' saved [95406/95406]
```

- Una vez tenemos descargado el paquete, vamos a tratar de instalarlo en nuestra máquina virtual. Para ello ejecutamos el siguiente comando: *dpkg -i pound\_2.7-1.3\_amd64.deb*

```
root@m3:/home/miguel444# dpkg -i pound_2.7-1.3_amd64.deb
Seleccionando el paquete pound previamente no seleccionado.
(Leyendo la base de datos ... 74385 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) ...
root@m3:/home/miguel444#
```

- Una vez finalizada la instalación del balanceador, vamos a proceder a configurarlo para poder redirigir las peticiones a las máquinas M1 y M2 que forman nuestra granja web. Para ello debemos modificar el fichero */etc/pound/pound.cfg* y añadir lo siguiente:

```
GNU nano 2.9.3 /etc/pound/pound.cfg
## redirect all requests on port 8080 ("ListenHTTP") to the local webserver (see "Service" below):
ListenHTTP
    Address 192.168.56.104
    Port    80

    Service
        Backend
            Address 192.168.56.12
            Port    80
        End
        Backend
            Address 192.168.56.102
            Port    80
        End
    End
End
```

- Address: IP de la máquina balanceadora
- Creamos dos Backend para cada servidor final y especificamos sus correspondientes direcciones IP y el puerto de escucha.
- Una vez instalado y configurado el balanceador vamos a reiniciar el servicio, pero antes debemos modificar el fichero */etc/default/pound* y poner la variable ‘startup’ con valor 1 para que el servicio se ponga en marcha:

```
GNU nano 2.9.3 /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 reiniciamos el servicio con *service pound restart* y ya podemos ver como nuestro balanceador distribuye las peticiones a los servidores correctamente (cabe destacar que previamente se han parado los servicios de nginx y haproxy para evitar problemas).

```
mapd0004@LAPTOP-9FUP9E1A:~$ curl http://192.168.56.104
<HTML>
<BODY>

PETICIÓN RECIBIDA POR LA MÁQUINA M2

</BODY>
</HTML>
mapd0004@LAPTOP-9FUP9E1A:~$ curl http://192.168.56.104
<HTML>
<BODY>

PETICIÓN RECIBIDA POR LA MÁQUINA M1

</BODY>
</HTML>
mapd0004@LAPTOP-9FUP9E1A:~$ curl http://192.168.56.104
<HTML>
<BODY>

PETICIÓN RECIBIDA POR LA MÁQUINA M2

</BODY>
</HTML>
mapd0004@LAPTOP-9FUP9E1A:~$ curl http://192.168.56.104
<HTML>
<BODY>

PETICIÓN RECIBIDA POR LA MÁQUINA M1

</BODY>
</HTML>
mapd0004@LAPTOP-9FUP9E1A:~$
```

Para terminar, vamos a probar el desempeño del balanceador con la herramienta de Apache Benchmark y así obtener unas medidas de rendimiento al igual que se ha hecho con los balanceadores explicados anteriores en la práctica.

```
mapd0004@LAPTOP-9FUP9E1A: ~
Server Software:      Apache/2.4.29
Server Hostname:      192.168.56.104
Server Port:          80

Document Path:        /index.html
Document Length:      70 bytes

Concurrency Level:    10
Time taken for tests:  118.391 seconds
Complete requests:    100000
Failed requests:       0
Total transferred:    33900000 bytes
HTML transferred:     7000000 bytes
Requests per second:  844.66 [#/sec] (mean)
Time per request:     11.839 [ms] (mean)
Time per request:     1.184 [ms] (mean, across all concurrent requests)
Transfer rate:        279.63 [Kbytes/sec] received

Connection Times (ms)
      min  mean[+/-sd] median   max
Connect:    0     0   5.8      0   1337
Processing:  1    11  22.5      8   1350
Waiting:    1     9  20.8      7   1350
Total:      1    12  23.3      9   1368
```