

Práctica 3

Balanceo de carga en un sitio web

1. Nginx como balanceador	2
Configurar solo round-robin	2
Configura round-robin ponderado	3
Apache Benchmark (Round Robin ponderado)	5
Configura round-robin avanzado	6
NGINX otras cuestiones más avanzadas	7
2. HAproxy como balanceador	8
Configurar solo round robin	8
Ejemplo de funcionamiento	8
Configurar round robin y ponderación	9
Ejemplo de funcionamiento	9
Configurar y describe varias opciones avanzadas	10
backend	10
Defaults	11
Tipos de balanceadores	12
Apache Benchmark (Round Robin ponderado)	13
2.1 Habilita módulo de estadísticas en HAproxy	14
3. Instala y configura Gobetween	16
Instalamos gobetween	16
Modificamos el fichero:	16
Activamos gobetween	17
Ejemplo de funcionamiento	18
Apache Benchmark (Round Robin ponderado)	20
4. Instala y configura Zevenet	22
Instalación	22
Apache Benchmark (Round Robin ponderado)	29
5. Instala y configura POUND	31
Instalamos pound	31
Apache Benchmark (Round Robin ponderado)	34
5. Análisis comparativo de distintos balanceados en base a la carga con AB	36
Ver imagen más grande	38
Conclusión	39
Imagen ampliada	40

1. Nginx como balanceador

Configurar solo round-robin

Como vemos en esta configuración básica va alternando entre la máquina 1 y 2.

```
m3-igmorillas [Corriendo] - Oracle VM VirtualBox
Dispositivos  Ayuda

upstream balanceo_usuarioUGR{
    server 192.168.56.57;
    server 192.168.56.103;
}

server {
    listen 80;
    server_name balanceador_usuarioUGR;

    access_log /var/log/nginx/balanceador_usuarioUGR.access.log;
    error_log /var/log/nginx/balanceador_usuarioUGR.error.log;
    root /var/www/;

    location /
    {
        proxy_pass http://balanceo_usuarioUGR;
        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 "";
    }
}
```

```
ignacio@ignacio-X550EA:~$ curl http://192.168.56.101/ejemplo.html
<HTML>
  <BODY>
    Maquina 1
    Web de ejemplo de igmorillas para SWAP
    Email: igmorillas@correo.ugr.es
  </BODY>
</HTML>
ignacio@ignacio-X550EA:~$ curl http://192.168.56.101/ejemplo.html
<HTML>
  <BODY>
    Maquina2
    Web de ejemplo de igmorillas para SWAP
    Email: igmorillas@correo.ugr.es
  </BODY>
</HTML>
```

Configura round-robin ponderado

Añadiendo en la línea correspondiente al servidor en upstream, el parámetro **wight**, estaremos dando peso a ese servidor. Por ejemplo en nuestro caso estamos diciendo que de cada 6 peticiones, 4 van a la máquina 1, y las otras 2 a la máquina 2.

```
upstream balanceo_usuarioUGR{
    server 192.168.56.101 wight=4;
    server 192.168.56.102 wight=2;
}

server{
    listen 80;
```

```
<BODY>
    Maquina2
    Web de ejemplo de igmorillas para SWAP
    Email: igmorillas@correo.ugr.es
</BODY>
</HTML>
ignacio@ignacio-X550EA:~$ curl http://192.168.56.103/ejemplo.html
<HTML>
    <BODY>
        Maquina 1
        Web de ejemplo de igmorillas para SWAP
        Email: igmorillas@correo.ugr.es
    </BODY>
</HTML>
ignacio@ignacio-X550EA:~$ curl http://192.168.56.103/ejemplo.html
<HTML>
    <BODY>
        Maquina 1
        Web de ejemplo de igmorillas para SWAP
        Email: igmorillas@correo.ugr.es
    </BODY>
</HTML>
ignacio@ignacio-X550EA:~$ curl http://192.168.56.103/ejemplo.html
<HTML>
    <BODY>
        Maquina2
        Web de ejemplo de igmorillas para SWAP
        Email: igmorillas@correo.ugr.es
    </BODY>
</HTML>
ignacio@ignacio-X550EA:~$ curl http://192.168.56.103/ejemplo.html
<HTML>
    <BODY>
        Maquina 1
        Web de ejemplo de igmorillas para SWAP
        Email: igmorillas@correo.ugr.es
    </BODY>
</HTML>
ignacio@ignacio-X550EA:~$ curl http://192.168.56.103/ejemplo.html
<HTML>
    <BODY>
        Maquina 1
```

Apache Benchmark (Round Robin ponderado)

Vamos a someter a sobrecarga para comprobar el rendimiento de nuestra granja web.

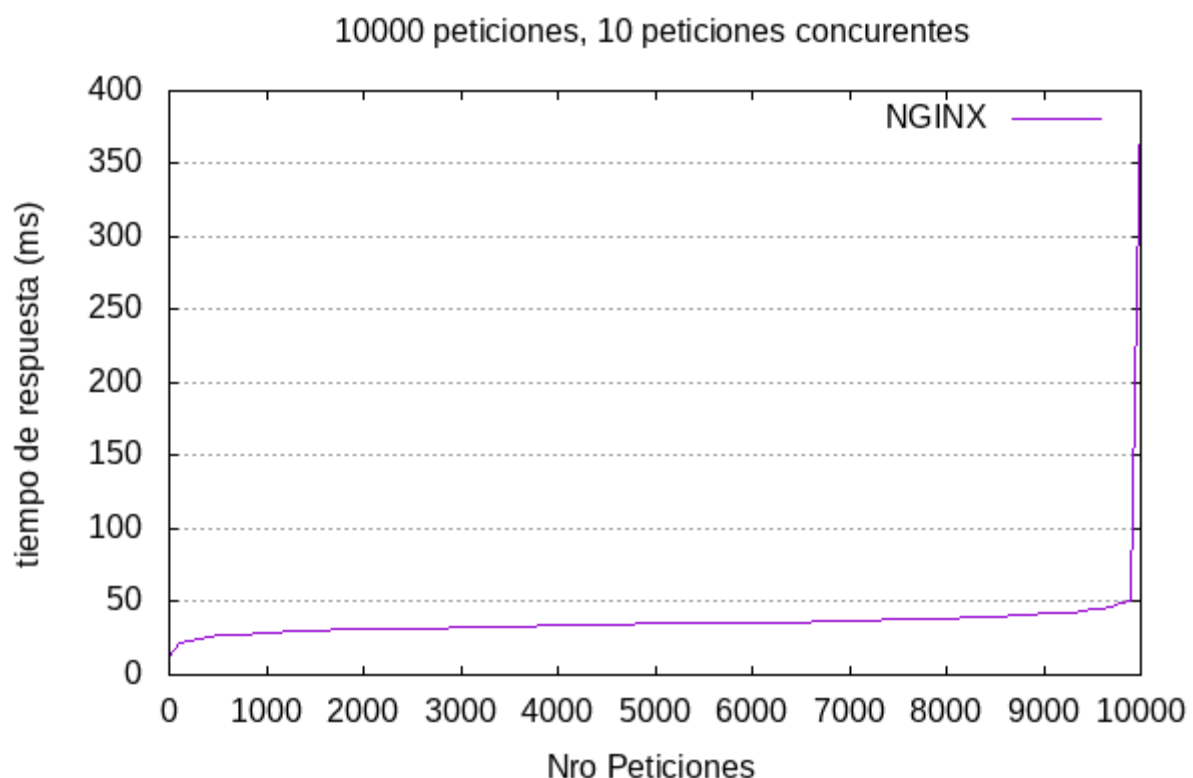
```
Server Software:      nginx/1.14.0
Server Hostname:      192.168.56.103
Server Port:          80

Document Path:        /ejemplo.html
Document Length:      118 bytes

Concurrency Level:     10
Time taken for tests:  38.707 seconds
Complete requests:     10000
Failed requests:       0
Total transferred:     3870000 bytes
HTML transferred:     1180000 bytes
Requests per second:   258.35 [#/sec] (mean)
Time per request:      38.707 [ms] (mean)
Time per request:      3.871 [ms] (mean, across all concurrent requests)
Transfer rate:         97.64 [Kbytes/sec] received

Connection Times (ms)
              min    mean[+/-sd] median    max
Connect:        0      1   0.7      1      9
Processing:     8     38  14.6     37    468
Waiting:        8     37  14.5     36    467
Total:          9     39  14.6     38    469

Percentage of the requests served within a certain time (ms)
 50%    38
 66%    41
 75%    43
 80%    44
 90%    48
 95%    52
 98%    55
 99%    59
100%   469 (longest request)
```



Configura round-robin avanzado

```
server 192.168.56.101 wight=4 max_fails=2 fail_timeout=30s
```

- **wight**: peso del servidor.
- **max_fails**: Número máximo de intentos de conexión.
- **fail_timeout**: indica el tiempo en el que deben ocurrir "max_fails" intentos fallidos de conexión para considerar al servidor no operativo.

Luego existen otros como:

- **down**: el servidor lo marca como caído.
- **backup**: el servidor solo se le pasa trafico si alguno de los otros servidores se cae o está ocupado.

```
upstream balanceo_usuarioUGR{
    server 192.168.56.101 weight=4 max_fails=2 fail_timeout=30s;
    server 192.168.56.102 weight=2 max_fails=5 fail_timeout=60s;
}

server{
    listen 80;
    server_name balanceador_usuarioUGR;

    access_log /var/log/nginx/balanceador_usuarioUGR.access.log;
    error_log /var/log/nginx/balanceador_usuarioUGR.error.log;
    root /var/www/;

    location /
    {
        proxy_pass http://balanceo_usuarioUGR;
        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 "";
    }
}
```

NGINX otras cuestiones más avanzadas

Existen además del balanceo round robin, y su variante ponderada, otros tipos de balanceo entre los que están:

- **least_conn:** la siguiente petición es atendida por el servidor con menos conexiones activas.

```
upstream balanceo_igmorillas{  
    least_conn;  
    server 192.168.56.101 wight=4 max_fails=2 fail_timeout=30s;  
    server 192.168.56.102 wight=2 max_fails=2 fail_timeout=30s;  
}
```

- **ip_hash:** “todas las peticiones que vengan de la misma IP se dirijan a la misma máquina servidora final.
Su mayor inconveniente es que no realiza un reparto equilibrado ya que aunque exista un servidor desocupado no se le destina a este, produciendo una saturación y que algunos servidores tengan mucha carga mientras otros están muy livianos en trabajo.

```
upstream balanceo_igmorillas{  
    ip_hash;  
    server 192.168.56.101 wight=4 max_fails=2 fail_timeout=30s;  
    server 192.168.56.102 wight=2 max_fails=2 fail_timeout=30s;  
}
```

- **keepalive:** todas las peticiones que vengan de la misma IP se dirijan a la misma máquina servidora final. Mantiene la conexión durante unos segundos entre balanceador y servidor. Soluciona el problema ya que una vez transcurrido este tiempo puede ser asignado a otro servidor.

```
upstream balanceo_igmorillas{  
    server 192.168.56.101 wight=4 max_fails=2 fail_timeout=30s;  
    server 192.168.56.102 wight=2 max_fails=2 fail_timeout=30s;  
    keepalive 3;  
}
```

2. HAproxy como balanceador

Configurar solo round robin

Básicamente las indicaciones del gui3n.

Para que funcione como Round Robin basta con a3adir la l3nea:

"balance roundrobin"

```
frontend http-in
    bind *:80
    default_backend balanceo_igmorillas

backend balanceo_igmorillas
    balance roundrobin
    server m1 192.168.56.101:80 maxconn 32
    server m2 192.168.56.103:80 maxconn 32
```

Ejemplo de funcionamiento

Con curl hemos estado realizando peticiones a la m3quina 3 y vemos que se est3an repartiendo entre las 2 m3quinas.

```
Maquina 1
Web de ejemplo de igmorillas para SWAP
Email: igmorillas@correo.ugr.es
</BODY>
</HTML>
ignacio@ignacio-X550EA:~$ curl http://192.168.56.102:80/ejemplo.html
<HTML>
    <BODY>
        Maquina2
        Web de ejemplo de igmorillas para SWAP
        Email: igmorillas@correo.ugr.es
    </BODY>
</HTML>
ignacio@ignacio-X550EA:~$ curl http://192.168.56.102:80/ejemplo.html
<HTML>
    <BODY>
        Maquina 1
        Web de ejemplo de igmorillas para SWAP
        Email: igmorillas@correo.ugr.es
    </BODY>
</HTML>
ignacio@ignacio-X550EA:~$ curl http://192.168.56.102:80/ejemplo.html
<HTML>
    <BODY>
        Maquina2
        Web de ejemplo de igmorillas para SWAP
        Email: igmorillas@correo.ugr.es
    </BODY>
</HTML>
ignacio@ignacio-X550EA:~$ curl http://192.168.56.102:80/ejemplo.html
<HTML>
    <BODY>
        Maquina 1
        Web de ejemplo de igmorillas para SWAP
```

Configurar round robin y ponderación

Para darle una ponderación basta con añadir en la línea de cada server el peso que le queramos dar.

```
frontend http-in
    bind *:80
    default_backend balanceo_igmorillas

backend balanceo_igmorillas
    balance roundrobin
    server m1 192.168.56.101:80 maxconn 32 weight 4
    server m2 192.168.56.103:80 maxconn 32 weight 2
```

Ejemplo de funcionamiento

Con curl hemos realizado 6 peticiones a la máquina 3 y vemos que 4 han ido a la máquina 1 y otras 2 han ido a la máquina 2. Por tanto, de cada 6 peticiones 4 irán a la primera máquina y 2 a la segunda.

```
ignacio@ignacio-X550EA:~$ curl http://192.168.56.102:80/ejemplo.html
<HTML>
  <BODY>
    Maquina 1
    Web de ejemplo de igmorillas para SWAP
    Email: igmorillas@correo.ugr.es
  </BODY>
</HTML>
ignacio@ignacio-X550EA:~$ curl http://192.168.56.102:80/ejemplo.html
<HTML>
  <BODY>
    Maquina2
    Web de ejemplo de igmorillas para SWAP
    Email: igmorillas@correo.ugr.es
  </BODY>
</HTML>
ignacio@ignacio-X550EA:~$ curl http://192.168.56.102:80/ejemplo.html
<HTML>
  <BODY>
    Maquina 1
    Web de ejemplo de igmorillas para SWAP
    Email: igmorillas@correo.ugr.es
  </BODY>
</HTML>
ignacio@ignacio-X550EA:~$ curl http://192.168.56.102:80/ejemplo.html
<HTML>
  <BODY>
    Maquina2
    Web de ejemplo de igmorillas para SWAP
    Email: igmorillas@correo.ugr.es
  </BODY>
</HTML>
ignacio@ignacio-X550EA:~$ curl http://192.168.56.102:80/ejemplo.html
<HTML>
  <BODY>
    Maquina 1
    Web de ejemplo de igmorillas para SWAP
    Email: igmorillas@correo.ugr.es
  </BODY>
</HTML>
ignacio@ignacio-X550EA:~$ curl http://192.168.56.102:80/ejemplo.html
<HTML>
  <BODY>
    Maquina 1
    Web de ejemplo de igmorillas para SWAP
```


Configurar y describe varias opciones avanzadas

```
frontend http-in
  bind *:80
  default_backend balanceo_igmorillas

backend balanceo_igmorillas
  balance roundrobin
  mode http
  option log-health-checks
  server m1 192.168.56.101:80 maxconn 32 weight 4 check fall 2 inter 1s
  server m2 192.168.56.103:80 maxconn 32 weight 2 check fall 2 inter 1s
```

backend

- **maxconn:** Máximo de número de conexiones permitidas al servidor.
- **weight:** Peso de cada servidor, las peticiones se reparten en función del peso del servidor. El servidor con mayor ponderación recibirá más peticiones que el resto.
- **check:** Comprueba el estado del servidor
- **<IP>:<puerto>:** reenvía a la IP y puerto señalados
- **option log-health-checks:** Puede registrar cualquier cambio en el estado de verificación o la salud del servidor.
Los registros pueden mostrar que un servidor falló verificaciones ocasionales antes de fallar, por ejemplo cuando:
 - No devuelve un estado HTTP válido
 - El puerto comienza a rechazar conexiones
 - El servidor deja de responder por completo.
- **fall 2 inter 1s:** HAProxy comprobará si el servidor Apache está disponible cada segundo y lo considerará muerto después de 2 comprobaciones fallidas consecutivas.
- **mode http:** acepta http tráfico en el puerto 80 para cualquier dirección IP con un máximo de 20480 conexiones.

Defaults

```
defaults
  timeout client 30s
  timeout connect 4s
  timeout server 30s
  timeout http-request 10s
  timeout http-keep-alive 2s
  timeout queue 5s
  timeout tunnel 2m
  timeout client-fin 1s
  timeout server-fin 1s
```

- **timeout client <timeout>**: establece el tiempo máximo de inactividad en el cliente
- **timeout connect <timeout>**: Establezca el tiempo máximo de espera para que se realice correctamente un intento de conexión con un servidor.
- **timeout server <timeout>**: Establezca el tiempo máximo de inactividad en el lado del servidor
- **timeout http-request <timeout>**: Establecer el tiempo máximo permitido para esperar una solicitud HTTP completa.
- **timeout http-keep-alive <timeout>**: Establezca el tiempo máximo permitido para esperar a que aparezca una nueva solicitud HTTP.
- **timeout queue <timeout>**: Establezca el tiempo máximo de espera en la cola para que una ranura de conexión esté libre.
- **timeout tunnel <timeout>**: Establezca el tiempo máximo de inactividad en el lado del cliente y del servidor para los túneles.
- **timeout client-fin <timeout>**: Establezca el tiempo de espera de inactividad en el lado del cliente para conexiones medio cerradas.
- **timeout server-fin <timeout>**: Establezca el tiempo de espera de inactividad en el lado del servidor para conexiones medio cerradas.

Tipos de balanceadores

Al igual que nginx tenemos distintos tipos de balanceadores, con el que hemos estado trabajando a sido round robin, pero están el: **Static-rr, Least connections, First, Source, URI, URL parameter, HDR,rdp-cookie.**

1. Least connections

La solicitud se envía al servidor con el menor número de conexiones activas.

```
backend balanceo_igmorillas
    balance leastconn
```

2. First

El primero que quede libre se le asignará la nueva solicitud empezando a comprobar siempre de arriba a abajo.

```
backend balanceo_igmorillas
    balance first
```

Podemos encontrar mas detalle en

<https://d2c.io/post/haproxy-load-balancer-part-2-backend-section-algorithms> , por que ya que algunos de ellos dependen del tipo de servidor, si son base de datos, o caches etc.

Apache Benchmark (Round Robin ponderado)

Vamos a someter a sobrecarga para comprobar el rendimiento de nuestra granja web.

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

Document Path:        /ejemplo.html
Document Length:      119 bytes

Concurrency Level:    10
Time taken for tests:  38.262 seconds
Complete requests:     10000
Failed requests:       3333
  (Connect: 0, Receive: 0, Length: 3333, Exceptions: 0)
Total transferred:    3886667 bytes
HTML transferred:     1186667 bytes
Requests per second:  261.35 [#/sec] (mean)
Time per request:     38.262 [ms] (mean)
Time per request:     3.826 [ms] (mean, across all concurrent requests)
Transfer rate:        99.20 [Kbytes/sec] received
```

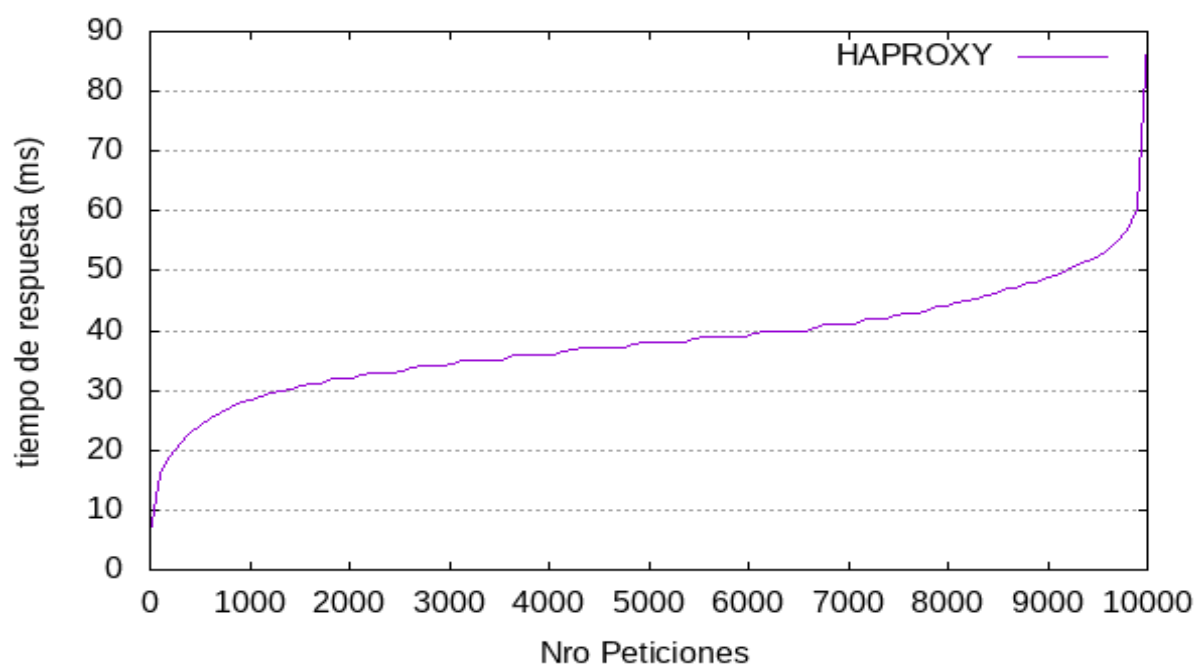
Connection Times (ms)

	min	mean[+/-sd]	median	max
Connect:	0	1 0.8	1	10
Processing:	5	37 8.5	37	89
Waiting:	5	37 8.5	37	89
Total:	6	38 8.5	38	89

Percentage of the requests served within a certain time (ms)

50%	38
66%	40
75%	43
80%	44
90%	49
95%	52
98%	57
99%	61
100%	89 (longest request)

10000 peticiones, 10 peticiones concurrentes



2.1 Habilita módulo de estadísticas en HAproxy

Añadimos y para que refresque cada 10s:

- **stats refresh 10s: Se refresca la página cada 10 segundos.**

```
global
    stats socket /var/lib/haproxy/stats
```

```
listen stats
    bind *:9999
    mode http
    stats enable
    stats uri /stats
    stats realm HAProxy\ Statistics
    stats auth igmorillas:igmorillas
    stats refresh 10s
```

Statistics Report for HAProxy x +

192.168.56.103:9999/stats

HAProxy version 1.8.8-1ubuntu0.11, released 2020/06/22

Statistics Report for pid 6390

> General process information

pid = 6390 (process #1, nbproc = 1, nbthread = 1)
uptime = 0d 0h01m25s
system limits: memmax = unlimited; ulimit.n = 4034
maxsock = 4034; maxconn = 2000; maxpipes = 0
current conns = 1; current pipes = 0/0; conn rate = 0/sec
Running tasks: 1/8; idle = 100 %

active UP
active UP, going down
active DOWN, going up
active or backup DOWN
active or backup DOWN for maintenance (MAINT)
active or backup SOFT STOPPED for maintenance

backup UP
backup UP, going down
backup DOWN, going up
not checked

Note: "NOLB"/"DRAIN" = UP with load-balancing disabled.

Display option:

- Scope:
- Hide **DOWN** servers
- Disable refresh
- Refresh now
- CSV export

External resources:

- Primary site
- Updates (v1.8)
- Online manual

stats		Queue			Session rate			Sessions							Bytes		Denied		Errors			Warnings		Server									
		Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle		
Frontend					0	1	-	1	1	2 000	2			1 246	38 491	0	0	0					OPEN			0	0	0	0	0			
Backend		0	0		0	0		0	0	200	0	0	0s	1 246	38 491	0	0		0	0	0	0	1m25s UP		0	0	0	0	0	0			

http-in		Queue			Session rate			Sessions							Bytes		Denied		Errors			Warnings		Server									
		Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle		
Frontend					0	2	-	0	1	2 000	7			630	2 588	0	0	0					OPEN										

balanceo_igmorillas		Queue			Session rate			Sessions							Bytes		Denied		Errors			Warnings		Server									
		Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle		
m1		0	0	-	0	2		0	1	32	5	5	11s	450	1 850		0		0	0	0	0	1m25s UP	L4OK in 1ms	4	Y	-	0	0	0s	-		
m2		0	0	-	0	1		0	1	32	2	2	13s	180	738		0		0	0	0	0	1m25s UP	L4OK in 1ms	2	Y	-	0	0	0s	-		
Backend		0	0		0	2		0	1	200	7	7	11s	630	2 588	0	0		0	0	0	0	1m25s UP		6	2	0	0	0	0s			

HAProxy version 1.8.8-1ubuntu0.11, released 2020/06/22

Statistics Report for pid 6390

> General process information

pid = 6390 (process #1, nbproc = 1, nbthread = 1)
 uptime = 0d 0h01m25s
 system limits: memmax = unlimited; ulimit-n = 4034
 maxsock = 4034; maxconn = 2000; maxpipes = 0
 current conns = 1; current pipes = 0/0; conn rate = 0/sec
 Running tasks: 1/8; idle = 100 %

active UP
 active UP, going down
 active DOWN, going up
 active or backup DOWN
 active or backup DOWN for maintenance (MAINT)
 active or backup SOFT STOPPED for maintenance
 backup UP
 backup UP, going down
 backup DOWN, going up
 not checked

Note: "NOLB"/"DRAIN" = UP with load-balancing disabled.

Display option:

- Scope :
- Hide 'DOWN' servers
- Disable refresh
- Refresh now
- CSV export

External resources:

- Primary site
- Updates (v1.8)
- Online manual

stats																															
	Queue			Session rate			Sessions						Bytes		Denied		Errors			Warnings		Server									
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle	
Frontend				0	1	-	1	1	2 000	2			1 246	38 491	0	0	0					OPEN									
Backend	0	0		0	0		0	0	200	0	0	0s	1 246	38 491	0	0		0	0	0	0	1m25s UP		0	0	0		0			

http-in																															
	Queue			Session rate			Sessions						Bytes		Denied		Errors			Warnings		Server									
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle	
Frontend				0	2	-	0	1	2 000	7			630	2 588	0	0	0					OPEN									

balanceo_igmorillas																														
	Queue			Session rate			Sessions						Bytes		Denied		Errors			Warnings		Server								
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle
m1	0	0	-	0	2		0	1	32	5	5	11s	450	1 850		0		0	0	0	0	1m25s UP	L4OK in 1ms	4	Y	-	0	0	0s	-
m2	0	0	-	0	1		0	1	32	2	2	13s	180	738		0		0	0	0	0	1m25s UP	L4OK in 1ms	2	Y	-	0	0	0s	-
Backend	0	0		0	2		0	1	200	7	7	11s	630	2 588	0	0		0	0	0	0	1m25s UP		6	2	0		0	0s	

3. Instala y configura Gobetween

Instalamos gobetween

```
sudo snap install gobetween --edge
```

Modificamos el fichero:

/var/snap/gobetween/common/gobetween.toml

Modificamos añadiendo:

Nota: si ponemos: **balance = "roundrobin"** tendremos el algoritmo round robin, para que funcione el ponderado ponemos el **weight**.

```
[servers.sample]
bind = "192.168.56.103:80"
protocol = "tcp"
#balance = "roundrobin"
balance = "weight"

max_connections = 10000
client_idle_timeout = "2s"
backend_idle_timeout = "2s"
backend_connection_timeout = "2s"

[servers.sample.discovery]
kind = "static"
static_list = [
    "192.168.56.101:80 weight=4",
    "192.168.56.102:80 weight=2",
]

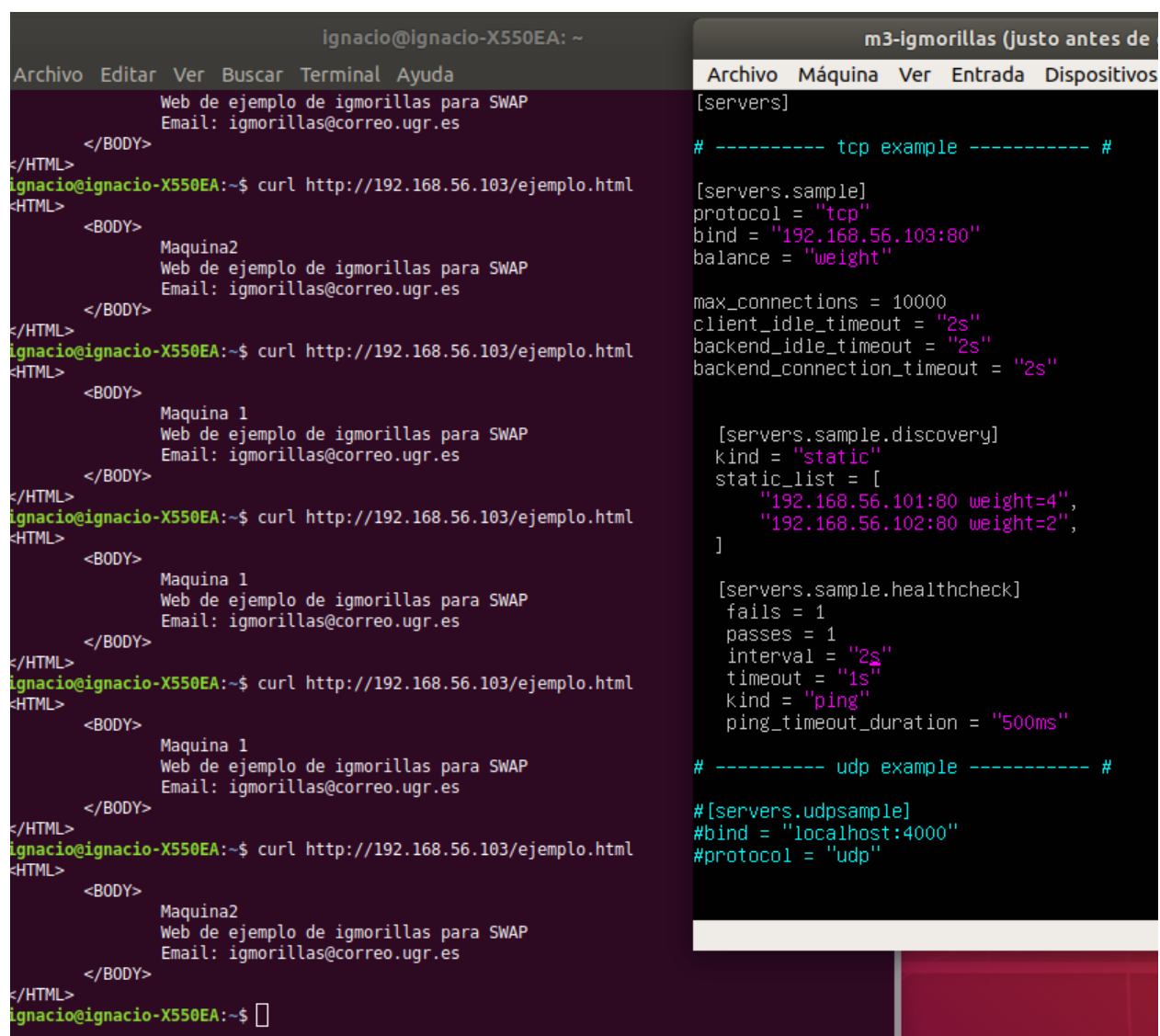
[servers.sample.healthcheck]
fails = 1
passes = 1
interval = "2s"
timeout="1s"
kind = "ping"
ping_timeout_duration = "500ms"
```

Activamos gobetween

```
sudo snap start gobetween
```

Ejemplo de funcionamiento

Realizamos peticiones con curl para comprobar el funcionamiento



The screenshot shows a terminal window with a dark background. The left pane displays the output of four curl requests to a web server. Each request returns an HTML response with a body containing information about a machine (Maquina 1 or Maquina 2) and a web example for SWAP. The right pane shows a configuration file for gobetween, titled 'm3-igmorillas (justo antes de)'. The configuration includes settings for TCP and UDP examples, server discovery, and health checks.

```
ignacio@ignacio-X550EA: ~  
Archivo Editar Ver Buscar Terminal Ayuda  
Web de ejemplo de igmorillas para SWAP  
Email: igmorillas@correo.ugr.es  
</BODY>  
</HTML>  
ignacio@ignacio-X550EA:~$ curl http://192.168.56.103/ejemplo.html  
<HTML>  
<BODY>  
Maquina2  
Web de ejemplo de igmorillas para SWAP  
Email: igmorillas@correo.ugr.es  
</BODY>  
</HTML>  
ignacio@ignacio-X550EA:~$ curl http://192.168.56.103/ejemplo.html  
<HTML>  
<BODY>  
Maquina 1  
Web de ejemplo de igmorillas para SWAP  
Email: igmorillas@correo.ugr.es  
</BODY>  
</HTML>  
ignacio@ignacio-X550EA:~$ curl http://192.168.56.103/ejemplo.html  
<HTML>  
<BODY>  
Maquina 1  
Web de ejemplo de igmorillas para SWAP  
Email: igmorillas@correo.ugr.es  
</BODY>  
</HTML>  
ignacio@ignacio-X550EA:~$ curl http://192.168.56.103/ejemplo.html  
<HTML>  
<BODY>  
Maquina 2  
Web de ejemplo de igmorillas para SWAP  
Email: igmorillas@correo.ugr.es  
</BODY>  
</HTML>  
ignacio@ignacio-X550EA:~$
```

```
m3-igmorillas (justo antes de  
Archivo Máquina Ver Entrada Dispositivos  
[servers]  
# ----- tcp example ----- #  
[servers.sample]  
protocol = "tcp"  
bind = "192.168.56.103:80"  
balance = "weight"  
max_connections = 10000  
client_idle_timeout = "2s"  
backend_idle_timeout = "2s"  
backend_connection_timeout = "2s"  
[servers.sample.discovery]  
kind = "static"  
static_list = [  
    "192.168.56.101:80 weight=4",  
    "192.168.56.102:80 weight=2",  
> ]  
[servers.sample.healthcheck]  
fails = 1  
passes = 1  
interval = "2s"  
timeout = "1s"  
kind = "ping"  
ping_timeout_duration = "500ms"  
# ----- udp example ----- #  
#[servers.udpsample]  
#bind = "localhost:4000"  
#protocol = "udp"
```


Apache Benchmark (Round Robin ponderado)

Vamos a someter a sobrecarga para comprobar el rendimiento de nuestra granja web.

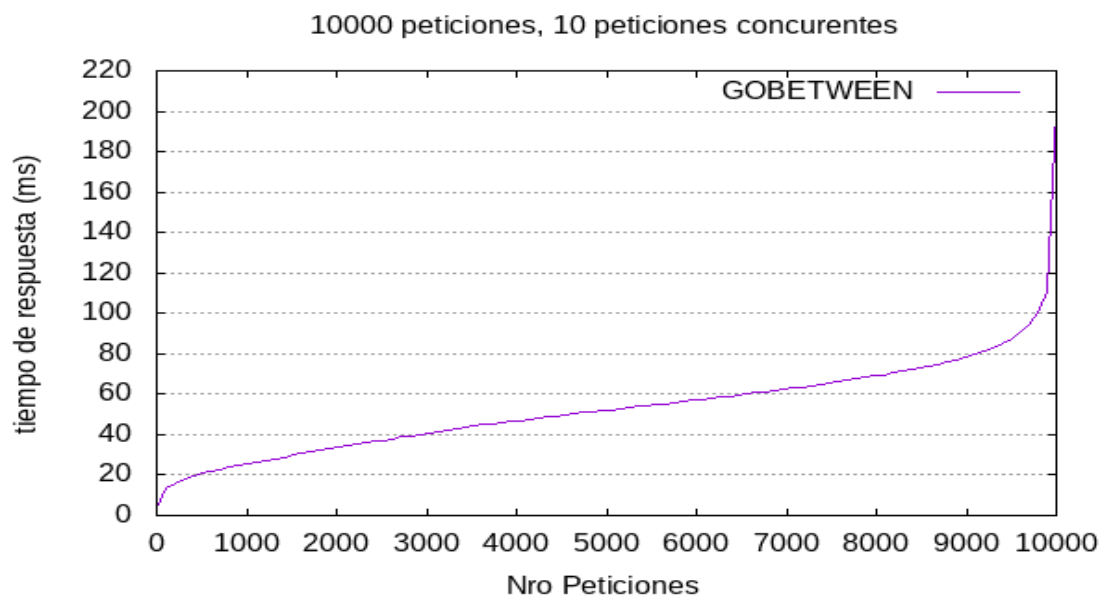
```
Server Software:      Apache/2.4.29
Server Hostname:      192.168.56.103
Server Port:          80

Document Path:        /ejemplo.html
Document Length:      119 bytes

Concurrency Level:    10
Time taken for tests:  52.689 seconds
Complete requests:    10000
Failed requests:       3382
  (Connect: 0, Receive: 0, Length: 3382, Exceptions: 0)
Total transferred:    3886618 bytes
HTML transferred:     1186618 bytes
Requests per second:  189.79 [#/sec] (mean)
Time per request:     52.689 [ms] (mean)
Time per request:     5.269 [ms] (mean, across all concurrent requests)
Transfer rate:        72.04 [Kbytes/sec] received

Connection Times (ms)
              min    mean[+/-sd] median    max
Connect:        0      1   0.6      1      9
Processing:      4     52  21.0     51    201
Waiting:        4     51  20.5     50    200
Total:          4     53  21.1     52    202

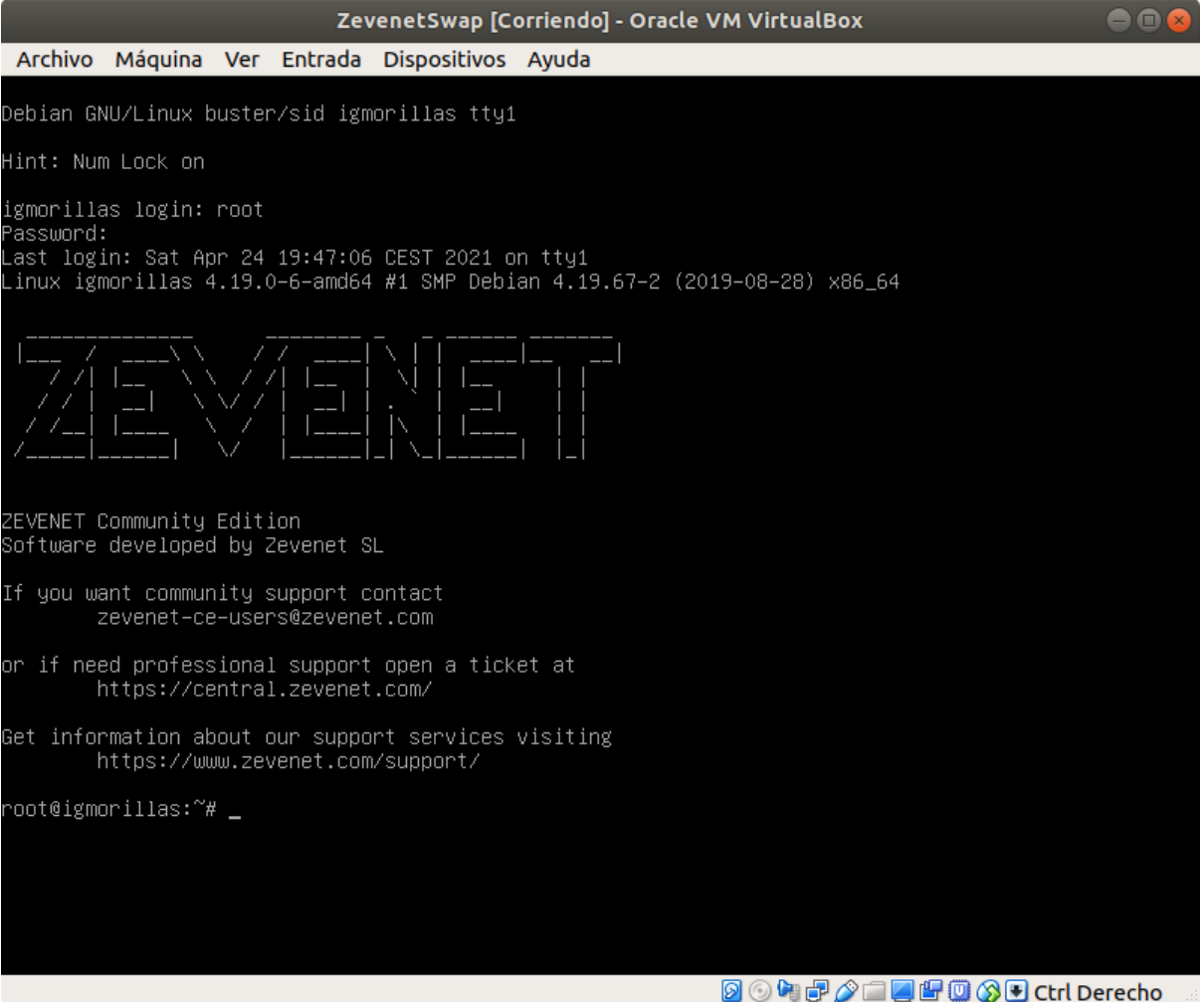
Percentage of the requests served within a certain time (ms)
 50%    52
 66%    60
 75%    65
 80%    69
 90%    78
 95%    88
 98%   100
 99%   110
100%   202 (longest request)
```



4. Instala y configura Zevenet

Instalación

- 1) Descargamos la imagen iso de: <https://www.zevenet.com/products/community/>
- 2) Y seguimos los pasos indicados :
<https://www.zevenet.com/knowledge-base/enterprise-edition/enterprise-edition-v3-04-administration-guide/enterprise-edition-v3-04-installation-guide/>
- 3) Iniciamos sesión con ID: root y PASS: Swap1234



```
Debian GNU/Linux buster/sid igmorillas tty1
Hint: Num Lock on

igmorillas login: root
Password:
Last login: Sat Apr 24 19:47:06 CEST 2021 on tty1
Linux igmorillas 4.19.0-6-amd64 #1 SMP Debian 4.19.67-2 (2019-08-28) x86_64

  _____
 /  _  _  _  \
|  _ \| | | | | | |
| |_) | | | | |
|  _ \| | | | |
|_| \_|_|_|_|_|

ZEVENET Community Edition
Software developed by Zevenet SL

If you want community support contact
zevenet-ce-users@zevenet.com

or if need professional support open a ticket at
https://central.zevenet.com/

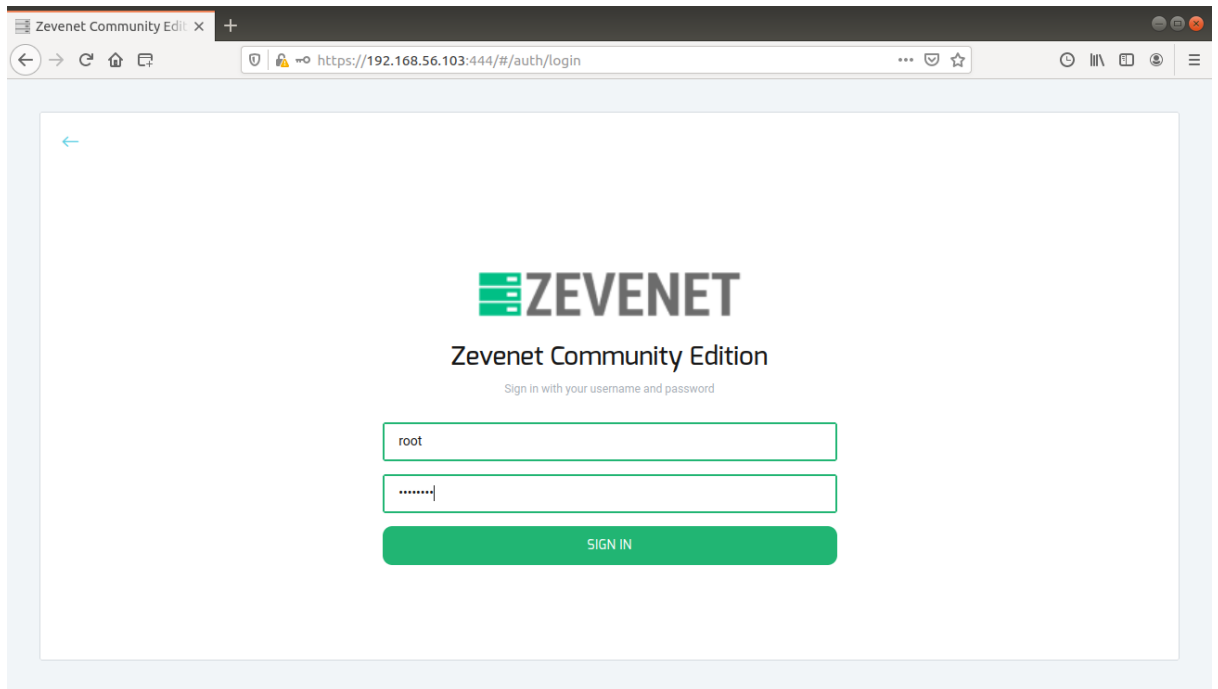
Get information about our support services visiting
https://www.zevenet.com/support/

root@igmorillas:~# _
```

- 4) Comprobamos que zevenet esté activa y la configuración de ifconfig está correcta.
- 5) Desde nuestro navegador accedemos a:

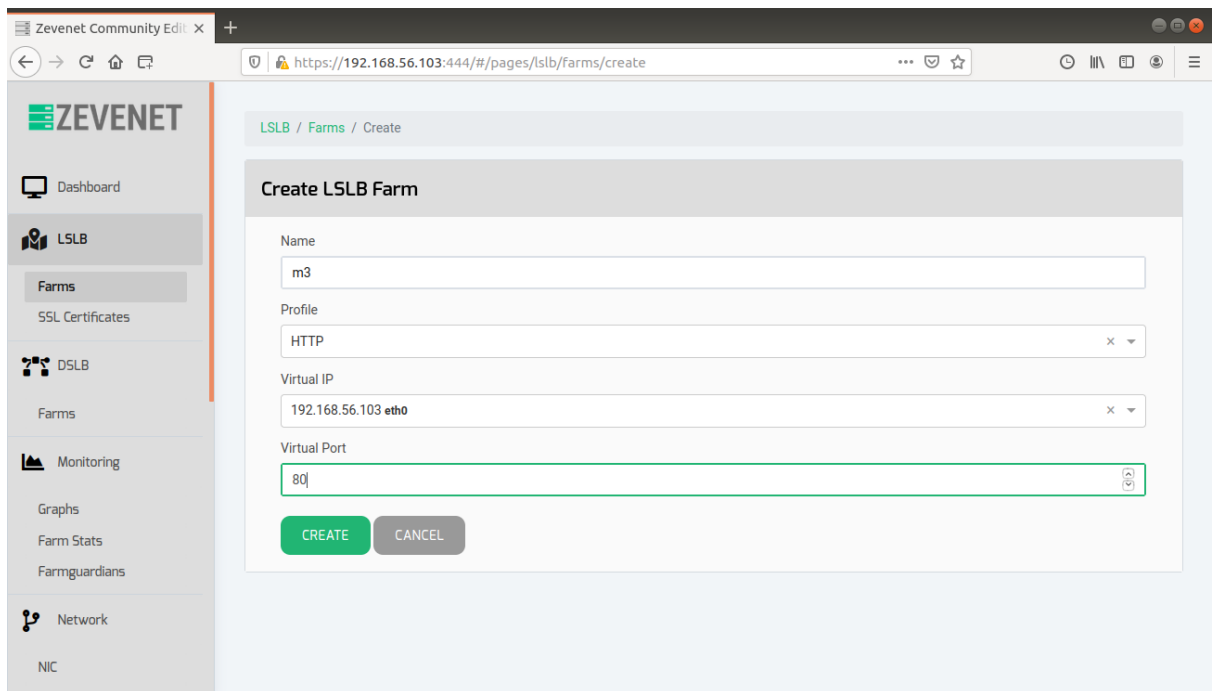
<https://192.168.56.103:444>

6) Nos logueamos con nuestra identificación anterior.



The screenshot shows a web browser window with the Zevenet Community Edition login page. The URL in the address bar is `https://192.168.56.103:444/#/auth/login`. The page features the Zevenet logo and the text "Zevenet Community Edition". Below the logo, it says "Sign in with your username and password". There are two input fields: the first contains the username "root", and the second contains a masked password ".....". A green "SIGN IN" button is located below the password field.

7) Creamos nuestra granja LSLB

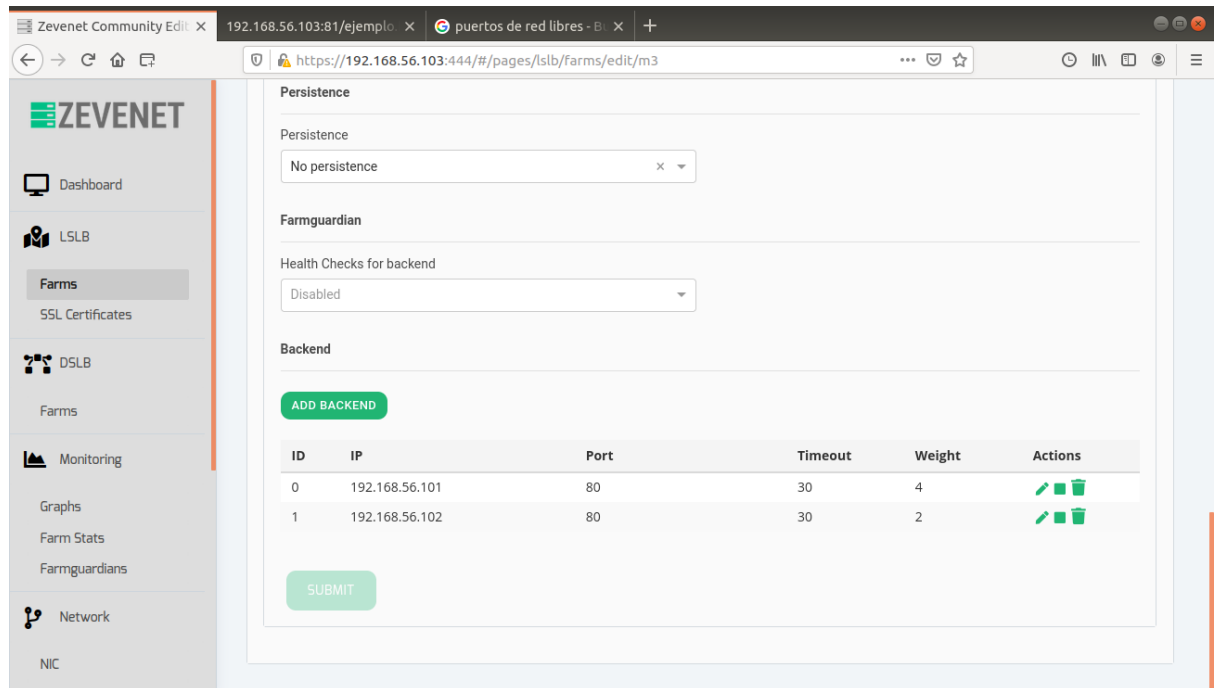


The screenshot shows the Zevenet Community Edition interface for creating a new LSLB farm. The left sidebar contains a navigation menu with options: Dashboard, LSLB, Farms, SSL Certificates, DSLB, Monitoring, Graphs, Farm Stats, Farmguardians, Network, and NIC. The main content area is titled "Create LSLB Farm" and contains the following fields:

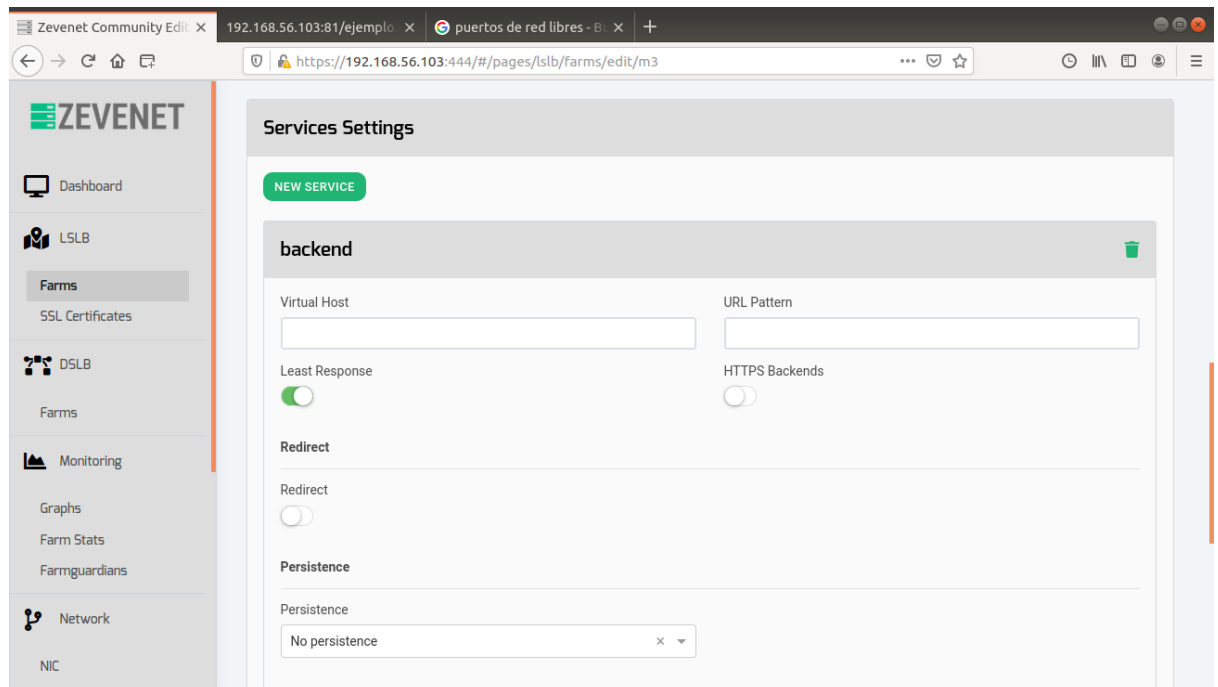
- Name:** A text input field containing "m3".
- Profile:** A dropdown menu set to "HTTP".
- Virtual IP:** A dropdown menu set to "192.168.56.103 eth0".
- Virtual Port:** A text input field containing "80".

At the bottom of the form, there are two buttons: "CREATE" (green) and "CANCEL" (grey).

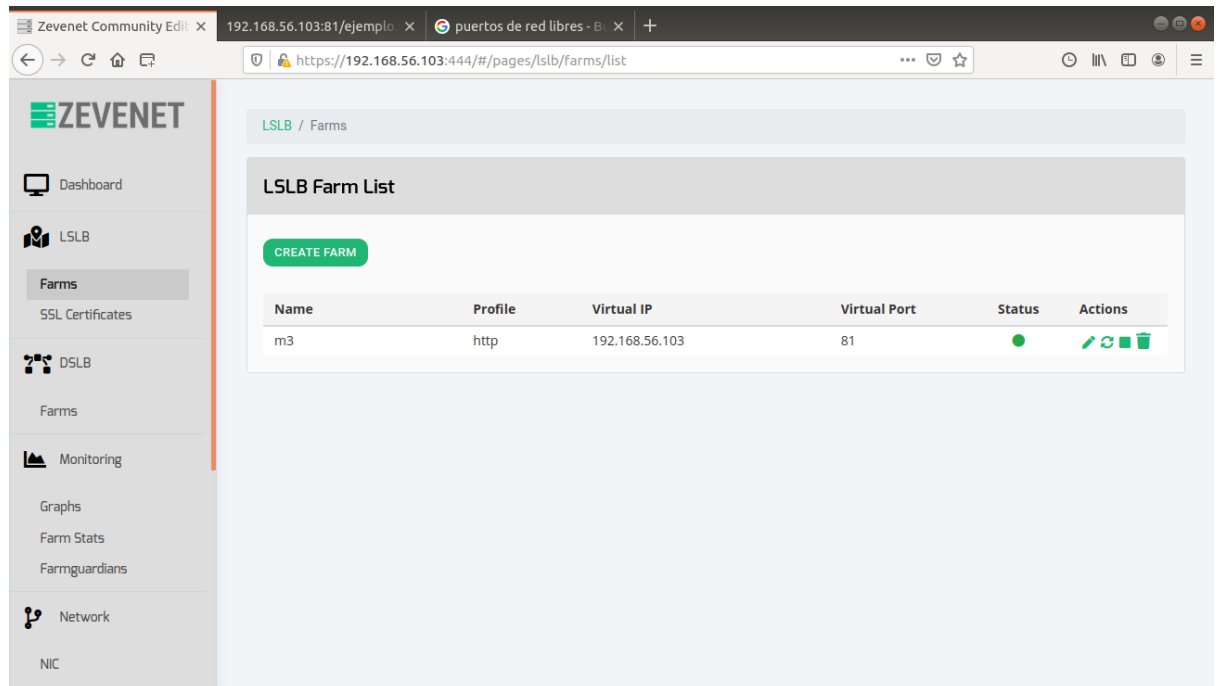
- 8) Una vez creada le damos a editar, una vez dentro al final de la página podremos añadir nuestros backend.



- 9) También marcamos la opción Round Robin, la que está junto a ella nos servirá si queremos implementar otro balanceador.



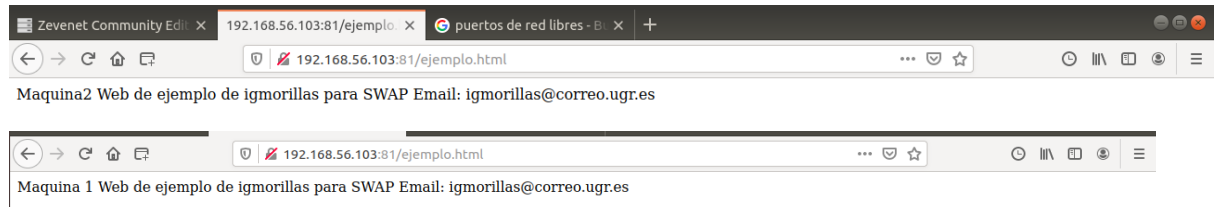
- 10) Si todo está correcto nos saldrá el estado de nuestra granja en verde que indicará que está operativa.



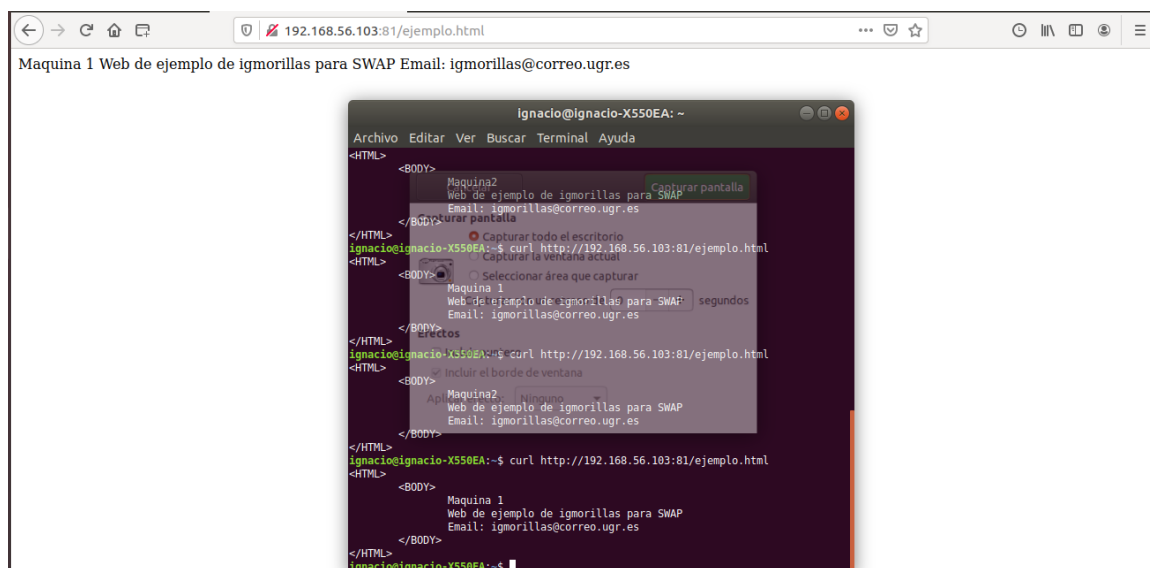
- 11) ahora desde nuestro navegador en otra pestaña, si accedemos a

<http://192.168.56.103:81/ejemplo.html>

vemos como se alternan los mensajes:



- 12) Desde la terminal igual comprobamos su funcionamiento:



Apache Benchmark (Round Robin ponderado)

Vamos a someter a sobrecarga para comprobar el rendimiento de nuestra granja web.

```
Server Software:      zproxy/0.1.6-0
Server Hostname:      192.168.56.103
Server Port:          80

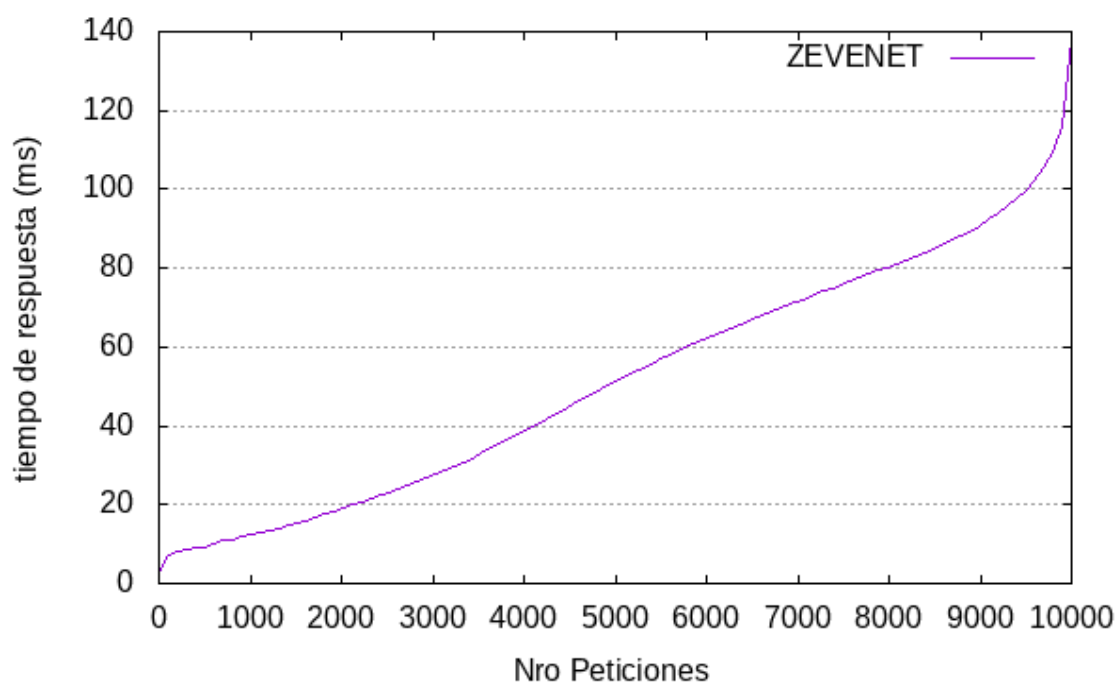
Document Path:        /ejemplo.html
Document Length:      55 bytes

Concurrency Level:    10
Time taken for tests:  29.301 seconds
Complete requests:    10000
Failed requests:      0
Non-2xx responses:    10000
Total transferred:    2170000 bytes
HTML transferred:     550000 bytes
Requests per second:  341.29 [#/sec] (mean)
Time per request:     29.301 [ms] (mean)
Time per request:     2.930 [ms] (mean, across all concurrent requests)
Transfer rate:        72.32 [Kbytes/sec] received

Connection Times (ms)
      min      mean[+/-sd] median    max
Connect:    0       3   2.5      4     13
Processing:  1      26  24.3     20    412
Waiting:    1      25  24.1     19    411
Total:      1      29  23.6     23    412

Percentage of the requests served within a certain time (ms)
 50%      23
 66%      32
 75%      39
 80%      44
 90%      59
 95%      74
 98%      87
 99%      95
100%     412 (longest request)
```

10000 peticiones, 10 peticiones concurrentes



5. Instala y configura POUND

Instalamos pound

```
sudo apt-get install -y pound
```

Pero por lo que he podido leer pound fue eliminado de ubuntu en febrero de 2018. El paquete no se mantenía y era incompatible con otros paquetes.

```
igmorillas@m3-igmorillas:~$ sudo apt-get install pound
[sudo] password for igmorillas:
Reading package lists... Done
Building dependency tree
Reading state information... Done
Package pound is not available, but is referred to by another package.
This may mean that the package is missing, has been obsoleted, or
is only available from another source

E: Package 'pound' has no installation candidate
igmorillas@m3-igmorillas:~$
```

Pero gracias a las indicaciones de un compañero en el apartado de cafetería de prado se puede instalar con:

```
# wget
http://kr.archive.ubuntu.com/ubuntu/pool/universe/p/pound/pound_2.6-6.1_amd64.deb

# sudo dpkg -i pound_2.6-6.1_amd64.deb
```

Para configurar modificamos el fichero: **/etc/pound/pound.cfg**

```
## redirect all requests on port 8080 ("ListenHTTP")
ListenHTTP
    Address 0.0.0.0
    Port    80

    ## allow PUT and DELETE also (by default disabled)
    xHTTP    0
    Client   30s

End
Service
    BackEnd
        Address 192.168.56.101
        Port    80
        Priority 4
    End
    BackEnd
        Address 192.168.56.102
        Port    80
        Priority 2
    End
End
```

Apache Benchmark (Round Robin ponderado)

Vamos a someter a sobrecarga para comprobar el rendimiento de nuestra granja web.

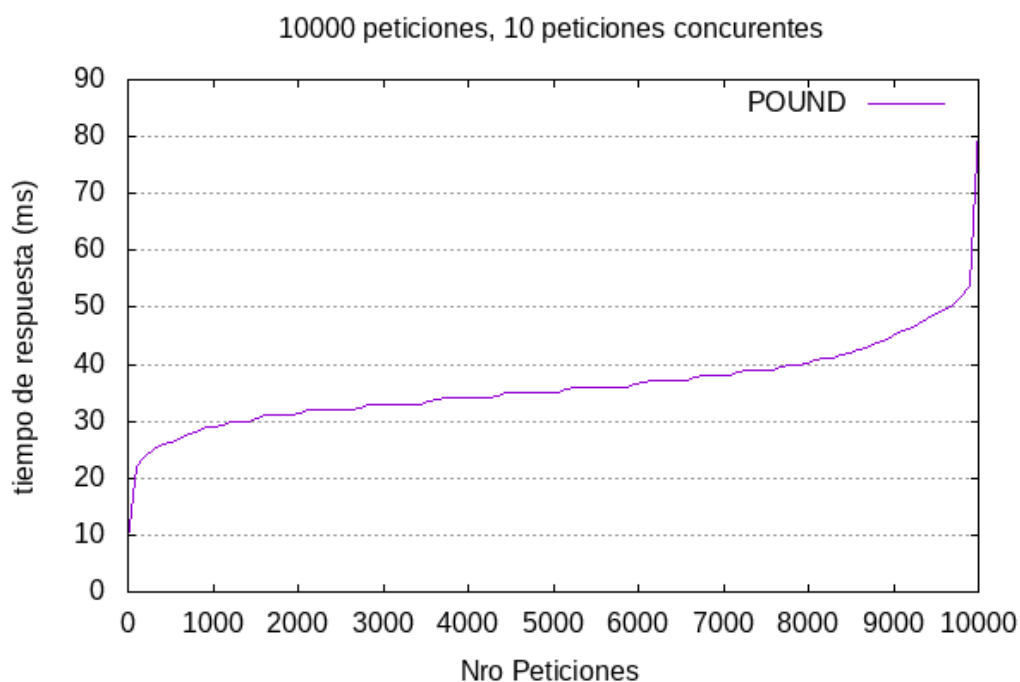
```
Server Software:      Apache/2.4.29
Server Hostname:      192.168.56.103
Server Port:          80

Document Path:        /ejemplo.html
Document Length:      118 bytes

Concurrency Level:     10
Time taken for tests:  36.298 seconds
Complete requests:     10000
Failed requests:       6667
  (Connect: 0, Receive: 0, Length: 6667, Exceptions: 0)
Total transferred:    3886667 bytes
HTML transferred:     1186667 bytes
Requests per second:   275.50 [#/sec] (mean)
Time per request:      36.298 [ms] (mean)
Time per request:      3.630 [ms] (mean, across all concurrent requests)
Transfer rate:         104.57 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median  max
Connect:        0    1  0.8      1    11
Processing:      7   35  8.6     35   106
Waiting:        7   35  8.6     35   106
Total:          8   36  8.6     36   106

Percentage of the requests served within a certain time (ms)
 50%    36
 66%    38
 75%    41
 80%    42
 90%    47
 95%    51
 98%    55
 99%    60
100%   106 (longest request)
```



5. Análisis comparativo de distintos balanceados en base a la carga con AB

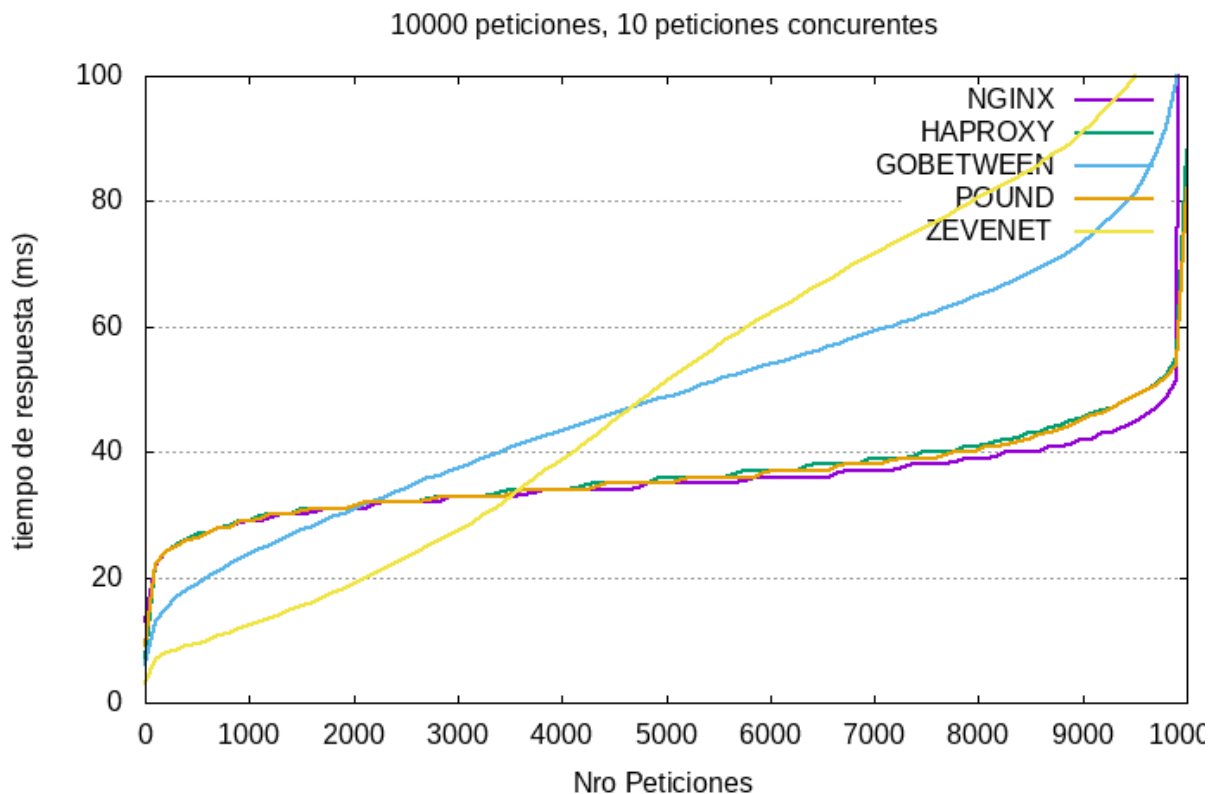
Balanceador	Algoritmo Balanceo	Tiempo(s)	Media de tiempo por petición(ms)
Nginx	Round Robin	38.707	3.871
HAProxy	Round Robin	38.262	3.826
Gobetween	Round Robin	52.689	5.269
Zevenet	Round Robin	29.301	2.930
POUND	Round Robin	36.298	3.630

Observando los datos vemos que el balanceador que mejor tiempo medio de respuesta ofrece es, de más rápido a menos:

Zevenet	2.930 s
POUND	3.630 s
HAProxy	3.826 s
Nginx	3.871 s
Gobetween	5.269 s

Zevenet es el que mejor tiempo de media ofrece, en cambio gobetween es el que mas retardo tiene.

Pero si observamos la gráfica siguiente



[Ver imagen más grande](#)

Observamos que aunque sea zevenet el que mejor tiempo de media da en los datos anteriores, observando la gráfica podemos deducir que:

- Para un número de peticiones bajo zevenet es mejor alternativa junto a haproxy, pero esto cambia cuando el número de peticiones están entorno 2000 - 3000 peticiones, que ya otros balanceadores nos ofrecen mejor rendimiento.
- El crecimiento del tiempo de respuesta con respecto al número de peticiones, para la mayoría de peticiones este crecimiento es logarítmico, pero para zevenet es mas bien un crecimiento que tiende mas a ser proporcional, mas lineal.
- Para los balanceadores Pound, Nginx y gobetween vemos que son muy similares sus gráficas y presentan el mismo tipo de crecimiento, pero podemos ver que para alta tasa de peticiones Nginx es la mejor, seguida de Pound y posteriormente por gobetween.

Conclusión

Podemos sacar en claro que para un número de peticiones bajo Zevenet es la mejor opción, mientras que para un número de peticiones media o alta, el mejor es Nginx

Imagen ampliada

