Swap Practica 3

Antonios Matzakos Chorianopoulos AH528893

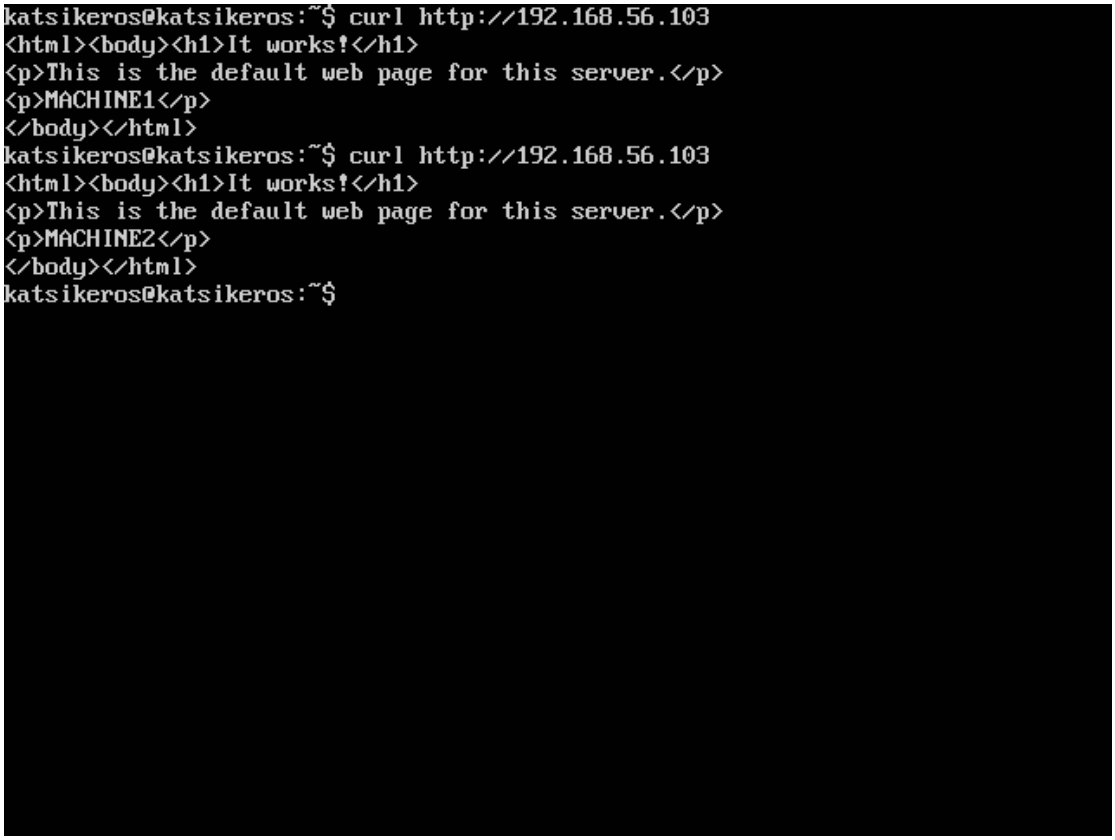
Theodoros Katsikeros AI015314

In this practice our purpose is to configure a network between various machines in order to have a balancer to share the load between the two machines .The first goal is to install the “nginx” (engine x) with the command: “ sudo apt-get install nginx” .

Then we create a file default.conf in the path /etc/nginx/conf.d

Once we are ready with the configuration we launch the nginx.

We call the balancer with curl from machine1 and as we see the machine1 appears first and then machine2.



Then , if we know that some machine is more powerful we can modify it in order to pass more traffic than the other. We do this through the modifier “weight”

```
 GNU nano 2.2.6              File: default.conf                        Modified

upstream apaches
{
        server 192.168.56.101 weight=1;
        server 192.168.56.102 weight=2;
}

server
{
        listen 80;
        server_name balanceador;

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

        location /
        {
                proxy_pass http://apaches;
                proxy_set_header Host $host;
                proxy_set_header X-Real-IP $remote_addr;
                proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
                proxy_http_version 1.1;
                proxy_set_header Connection "";
        }
}

^G Get Help  ^O WriteOut  ^R Read File ^Y Prev Page ^K Cut Text  ^C Cur Pos
^X Exit      ^J Justify   ^W Where Is  ^V Next Page ^U UnCut Text^T To Spell
```

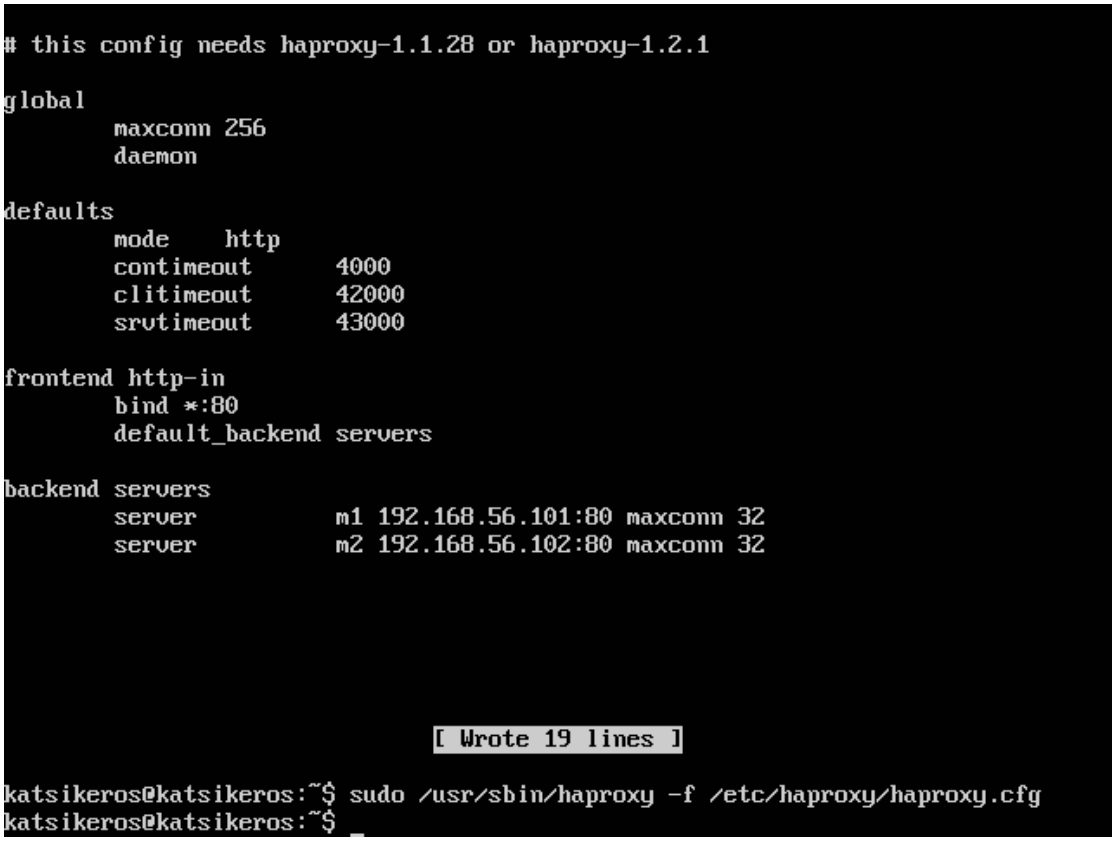
This way we say that Machine2 is two times “stronger” than Machine1 so it appears like Machine1, Machine2, Machine2.

```
katsikeros@katsikeros:~$ curl http://192.168.56.103
<html><body><h1>It works!</h1>
<p>This is the default web page for this server.</p>
<p>MACHINE1</p>
</body></html>
katsikeros@katsikeros:~$ curl http://192.168.56.103
<html><body><h1>It works!</h1>
<p>This is the default web page for this server.</p>
<p>MACHINE2</p>
</body></html>
katsikeros@katsikeros:~$ curl http://192.168.56.103
<html><body><h1>It works!</h1>
<p>This is the default web page for this server.</p>
<p>MACHINE2</p>
</body></html>
katsikeros@katsikeros:~$ _
```

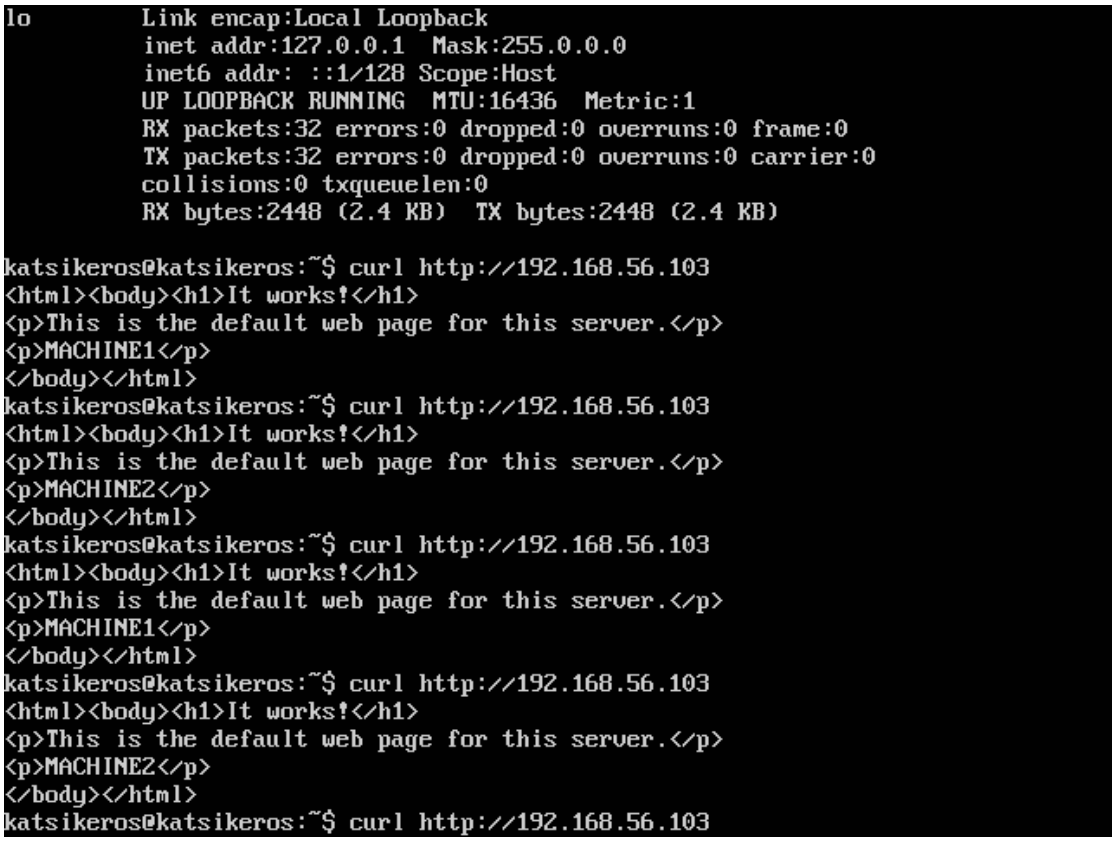
The next his we do is use the “ip_hash” command. With this one when the balancer thinks that one of the machines is down, it only allows the other one to be used. In this example it always chooses Machine2 because it never overloads.

```
<html><body><h1>It works!</h1>
<p>This is the default web page for this server.</p>
<p>MACHINE2</p>
</body></html>
katsikeros@katsikeros:~$ curl http://192.168.56.103
<html><body><h1>It works!</h1>
<p>This is the default web page for this server.</p>
<p>MACHINE2</p>
</body></html>
katsikeros@katsikeros:~$ curl http://192.168.56.103
<html><body><h1>It works!</h1>
<p>This is the default web page for this server.</p>
<p>MACHINE2</p>
</body></html>
katsikeros@katsikeros:~$ curl http://192.168.56.103
<html><body><h1>It works!</h1>
<p>This is the default web page for this server.</p>
<p>MACHINE2</p>
</body></html>
katsikeros@katsikeros:~$ curl http://192.168.56.103
<html><body><h1>It works!</h1>
<p>This is the default web page for this server.</p>
<p>MACHINE2</p>
</body></html>
katsikeros@katsikeros:~$ curl http://192.168.56.103
<html><body><h1>It works!</h1>
<p>This is the default web page for this server.</p>
<p>MACHINE2</p>
</body></html>
katsikeros@katsikeros:~$ _
```

Then we continue by installing haproxy and creating the config file /etc/haproxy/ haproxy.cfg as we did with nginx.



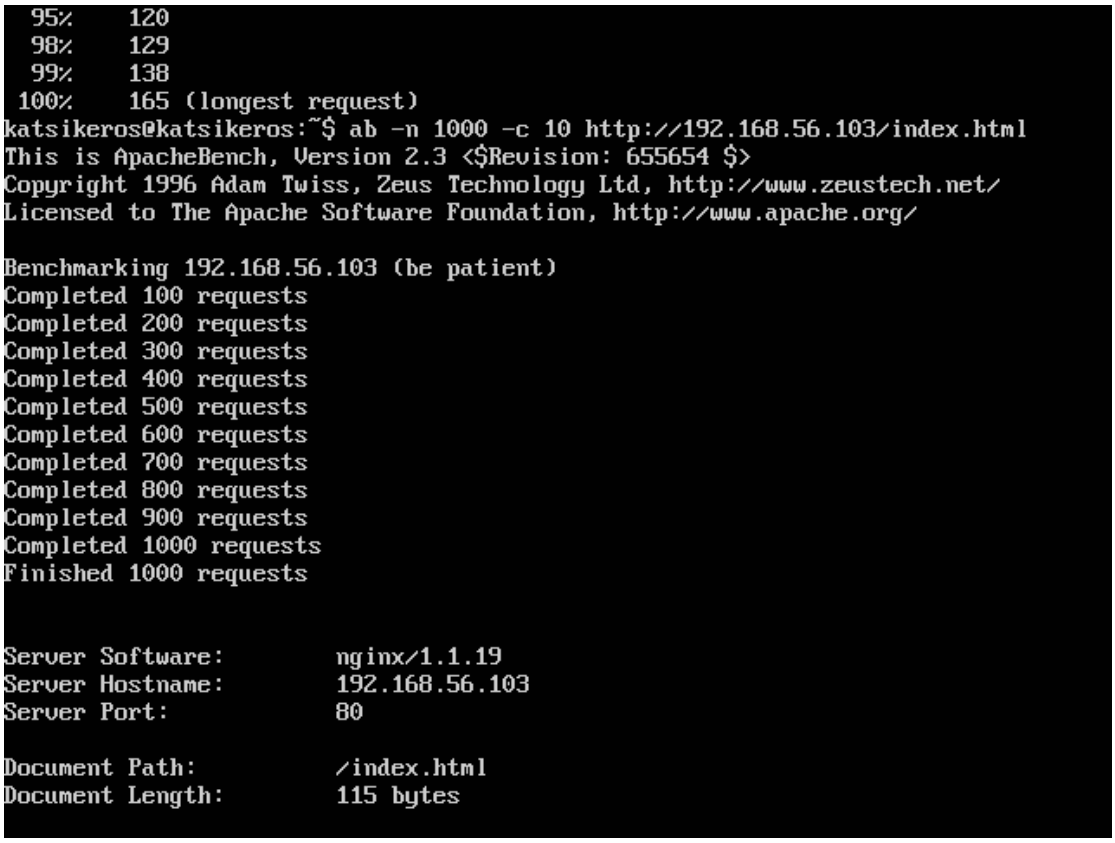
Again we use curl the same way as with nginx.



It appears alternatively Machine1, Machine2 as expected.

Lastly we run the benchmarks for nginx and for haproxy as we see in the screenshots below.

Nginx Banchmarks:



```
Concurrency Level:      10
Time taken for tests:   9.410 seconds
Complete requests:      1000
Failed requests:        0
Write errors:           0
Total transferred:      380000 bytes
HTML transferred:       115000 bytes
Requests per second:    106.27 [#/sec] (mean)
Time per request:       94.102 [ms] (mean)
Time per request:       9.410 [ms] (mean, across all concurrent requests)
Transfer rate:          39.44 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        2   24  17.1     21     171
Processing:    10   67  38.4     56     358
Waiting:        8   55  35.5     46     355
Total:         12   91  42.1     77     361

Percentage of the requests served within a certain time (ms)
  50%     77
  66%     96
  75%    108
  80%    116
  90%    141
  95%    169
  98%    231
  99%    262
 100%    361 (longest request)
katsikeros@katsikeros:~$ ~
```

Haproxy benchmarks:

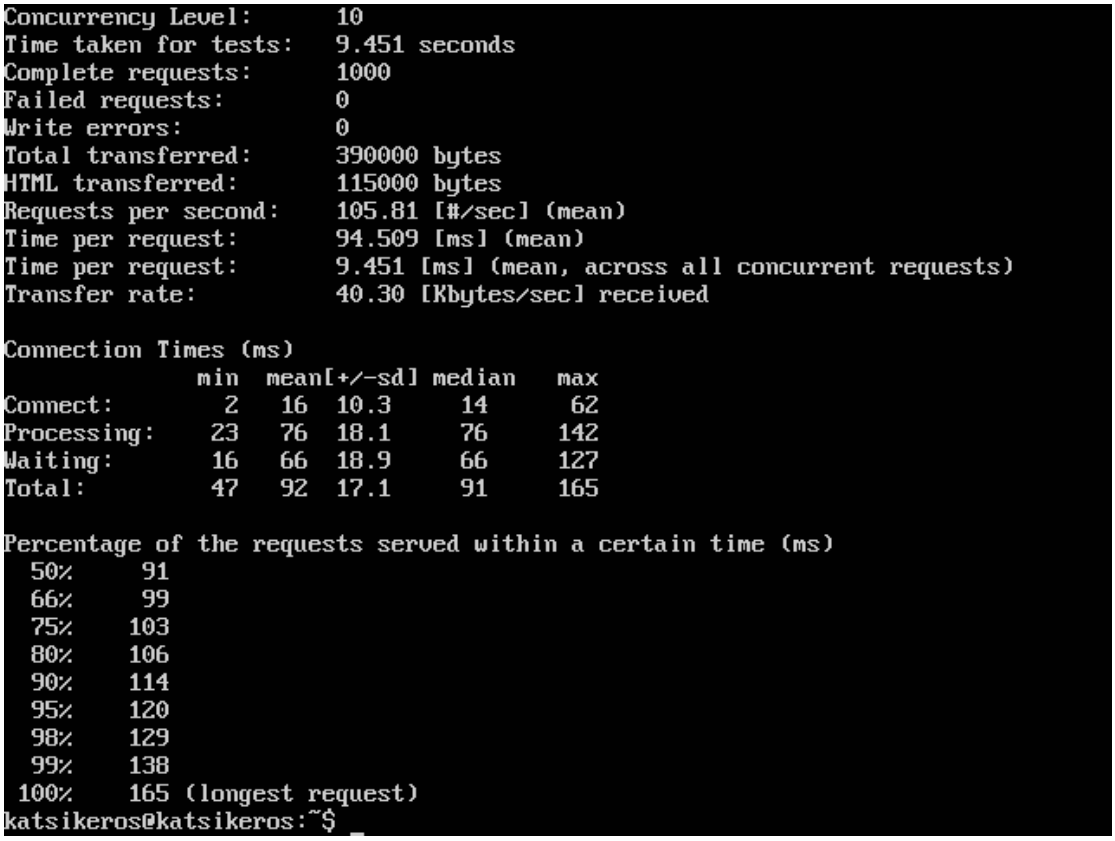
```
          collisions:0 txqueuelen:1000
          RX bytes:140360 (140.3 KB)  TX bytes:57950 (57.9 KB)

lo        Link encap:Local Loopback
katsikeros@katsikeros:~$ ab -n 1000 -c 10 http://192.168.56.103/index.html
This is ApacheBench, Version 2.3 <$Revision: 655654 $>
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 100 requests
Completed 200 requests
Completed 300 requests
Completed 400 requests
Completed 500 requests
Completed 600 requests
Completed 700 requests
Completed 800 requests
Completed 900 requests
Completed 1000 requests
Finished 1000 requests


Server Software:        Apache/2.2.22
Server Hostname:        192.168.56.103
Server Port:            80

Document Path:          /index.html
Document Length:        115 bytes
```



As we see at the beginning nginx is faster but at the end haproxy is way faster.