Laporan Final Project Pemrograman Jaringan

Anggota

- Sitti Chofifah (05111840000039) (Kelas E)
- Geizka Wahyu Fahriza (05111840000062) (Kelas E)
- Muhammad Iqbal Humam (05111840000103) (Kelas E)

Pendahuluan

Reverse proxy merupakan sebuah fitur/modul di dalam sebuah web server, yang berfungsi untuk melakukan *port forwarding* suatu *request*, dari *public request* menuju ke dalam sistem. Sebagai *proxy*, tentu juga memiliki kemampuan *content caching*, yaitu penyimpanan data sementara, agar saat terjadi request ulang, tidak perlu mengambil dari database. Dengan kemampuan-kemampuan tersebut, sebuah web server akan mampu mengerjakan pekerjaan-pekerjaan dari sebagai sekedar proxy, hingga sebagai *upstream proxy* yang kadang banyak difungsikan sebagai pengganti NAT. Dengan kemampuannya pula, reverse proxy akan mampu dimanfaatkan untuk menghemat IP Address.

Pada study case 1, reverse proxy digunakan untuk mengarahkan request ke server yang benar, maka request yang meminta gambar akan di arahkan ke server gamabr sedangkan request yang meminta pdf akan di arahkan ke server pdf dan semua request selain gambar dan pdf diarahkan ke server umum atau utama.

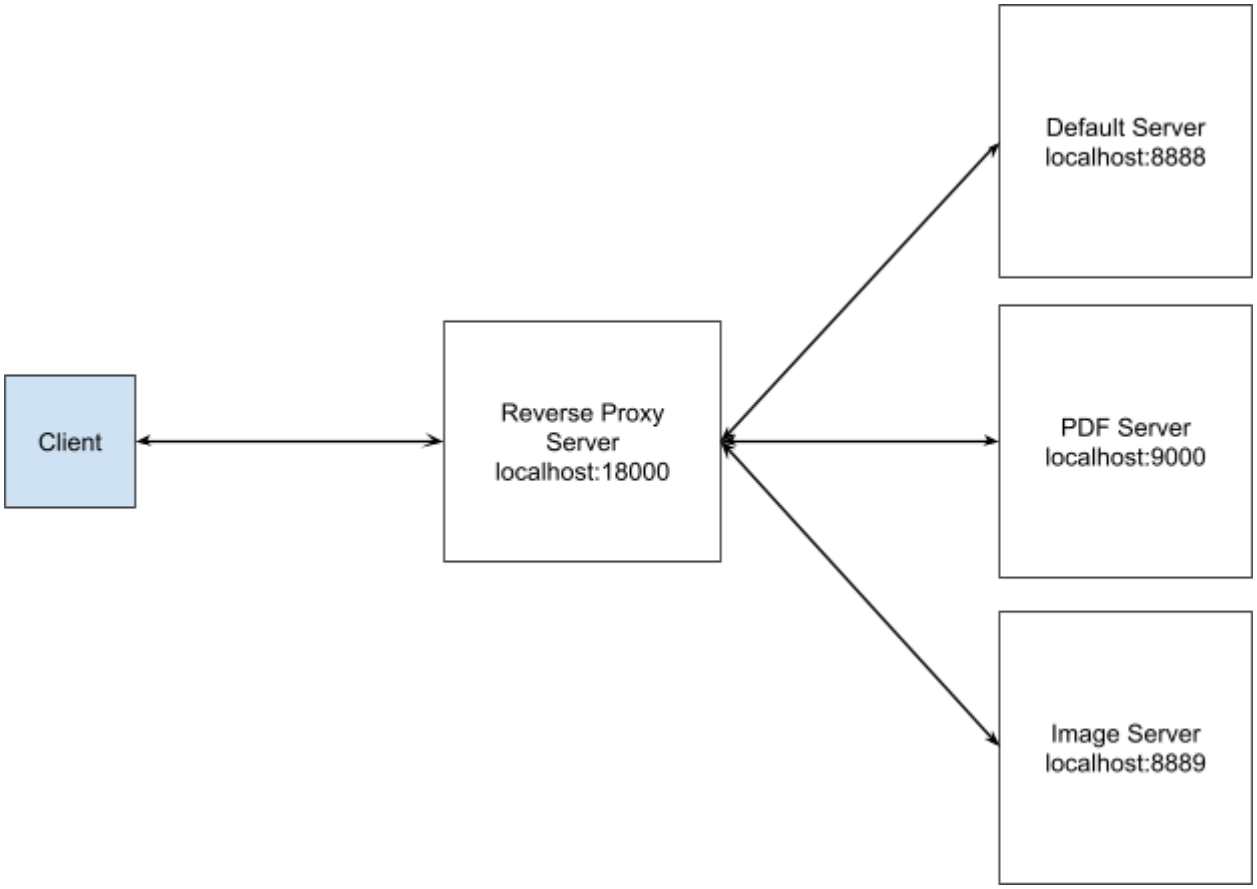
Pada study case 2, reverse proxy digunakan untuk membagi load ke banyak server, sehingga setiap server akan mendapatkan jumlah request yang sama dan akan lebih stabil.

Pembagian Pekerjaan

Nama	NRP	Pembagian Pekerjaan	Persentase
Sitti Chofifah	05111840000039	Membuat HTTP Server Image, PDF untuk Study Case 1	33%
Geizka Wahyu Fahriza	05111840000062	Membuat Reverse Proxy dan Load balancer untuk studi kasus 1 dan 2	33%
Muhammad Iqbal Humam	05111840000103	Membuat HTTP Server untuk Study Case 2	33%

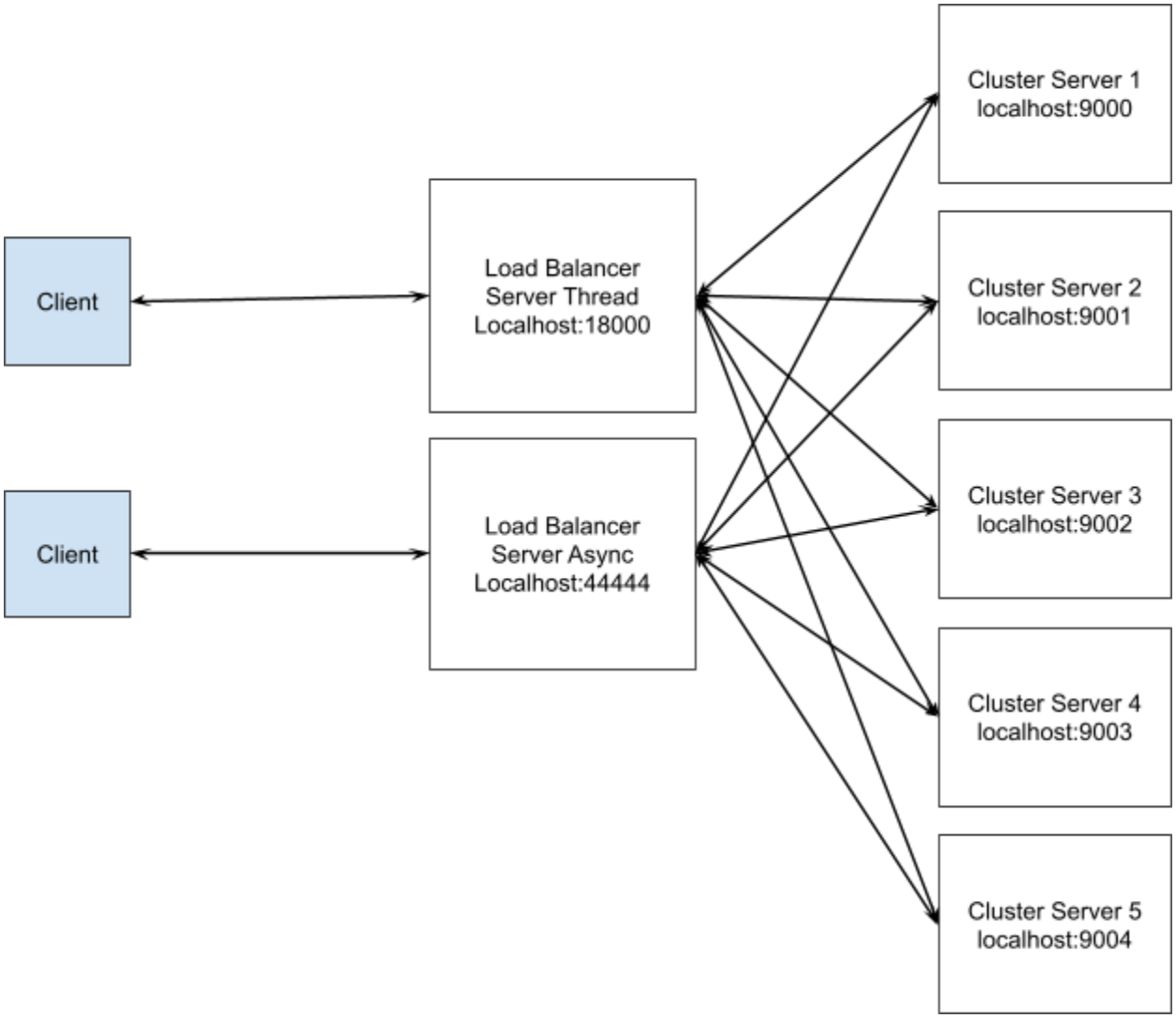
Arsitektur

Study Case 1



Pada studi kasus ke-1, terdapat reverse proxy pada port 18000 yang akan diakses oleh client. Reverse Proxy akan meneruskan request ke default server jika request bukan meminta gambar atau pdf (url tidak memiliki /images/ atau /pdf/). Sedangkan jika url memiliki path /pdf/ maka request akan diteruskan ke server pdf yang mempunyai port 9000. Jika url memiliki path /images/ maka request akan diteruskan ke server gambar yang mempunyai port 8889. Reverse Proxy berfungsi sebagai *middle man* sehingga pengguna dapat mengakses reverse proxy server saja untuk mendapatkan gambar, pdf, ataupun data lainnya.

Study Case 2



Pada studi kasus ke-2, terdapat 5 server dengan konfigurasi port 9000, 9001, 9002, 9003, 9004. Di mana kelima server tersebut akan meng-*handle request* yang diberikan dari dua server dengan dua metode *load balancer*, yaitu *async* dan *thread*. Dua metode ini bertujuan untuk membagi *request* ke sejumlah server yang sudah disiapkan untuk mencegah dan mengurangi *failed request.* Cara pembagian *request* ke lima server yang ada menggunakan metode Round Robin, di mana *request* akan dibagikan secara urut.

Pengujian

Study Case 1

Pada pengujiannya, *reverse proxy* diakses dari browser yang ada di desktop. Browser mengakses link http://localhost:18000, http://localhost:18000/images/,

http://localhost:18000/pdf/, http://localhost:18000/images/pokijan.jpg, http://localhost:18000/pdf/rfc2616.pdf

Study Case 2

Pada pengujiannya, *load balancer* diberikan 10.000 *request* dengan jumlah server aktif dan *concurrency* yang berbeda pula. Jumlah server aktif yang digunakan pada pengujian ini adalah 1, 2, 3, 4, dan 5. Sedangkan jumlah *concurrency* yang dipakai pada pengujian ini adalah 2, 5, dan 10.

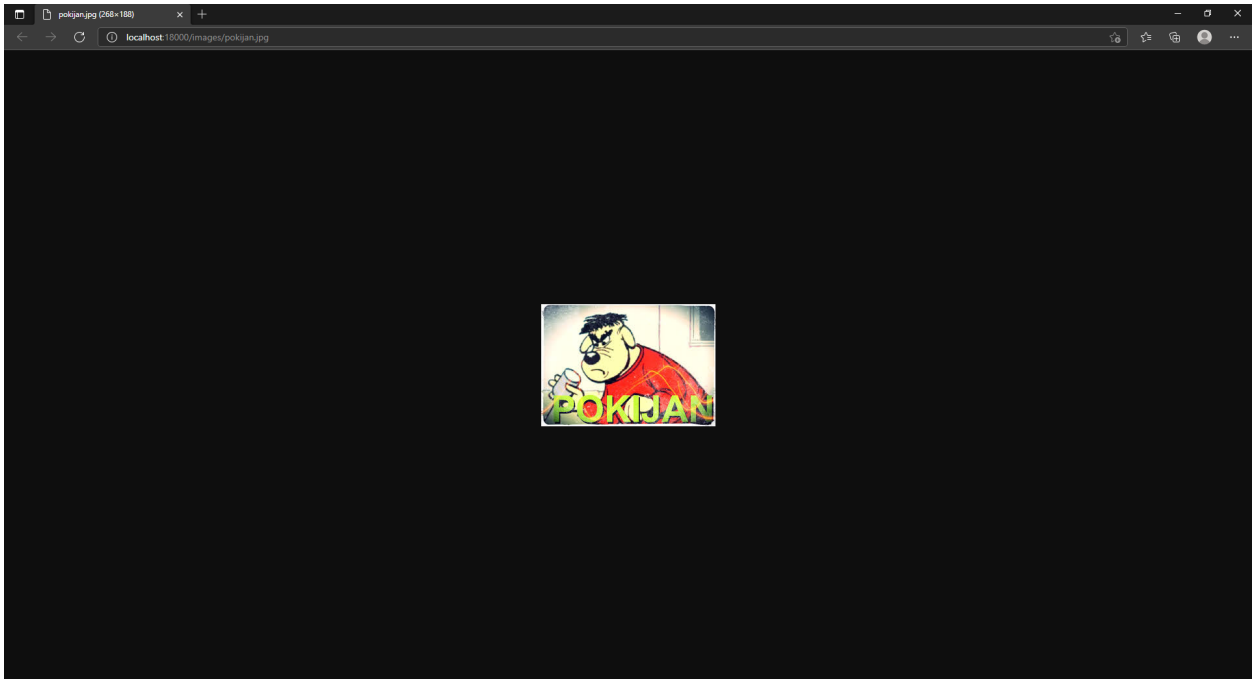
Screenshot Hasil

Study Case 1

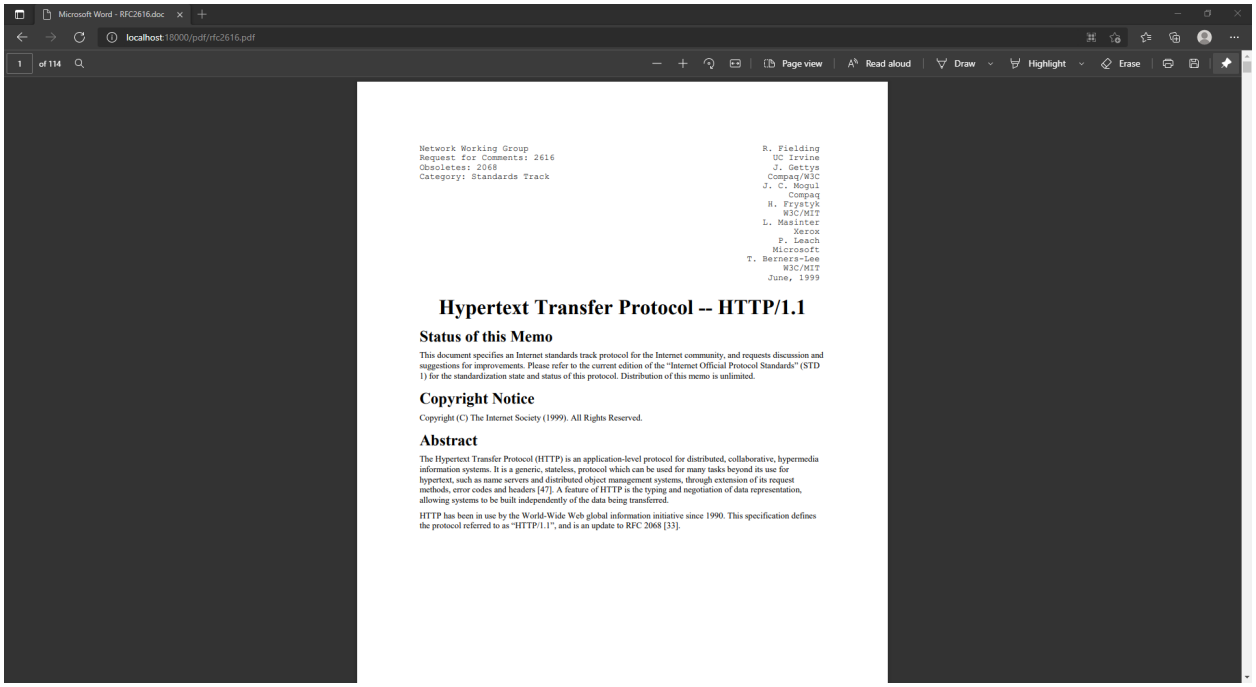
Berikut merupakan contoh screenshot testing reverse proxy menggunakan browser internet edge :

1. Membuka link url http://localhost:18000

2. Membuka link url http://localhost:18000/images/pokijan.jpg



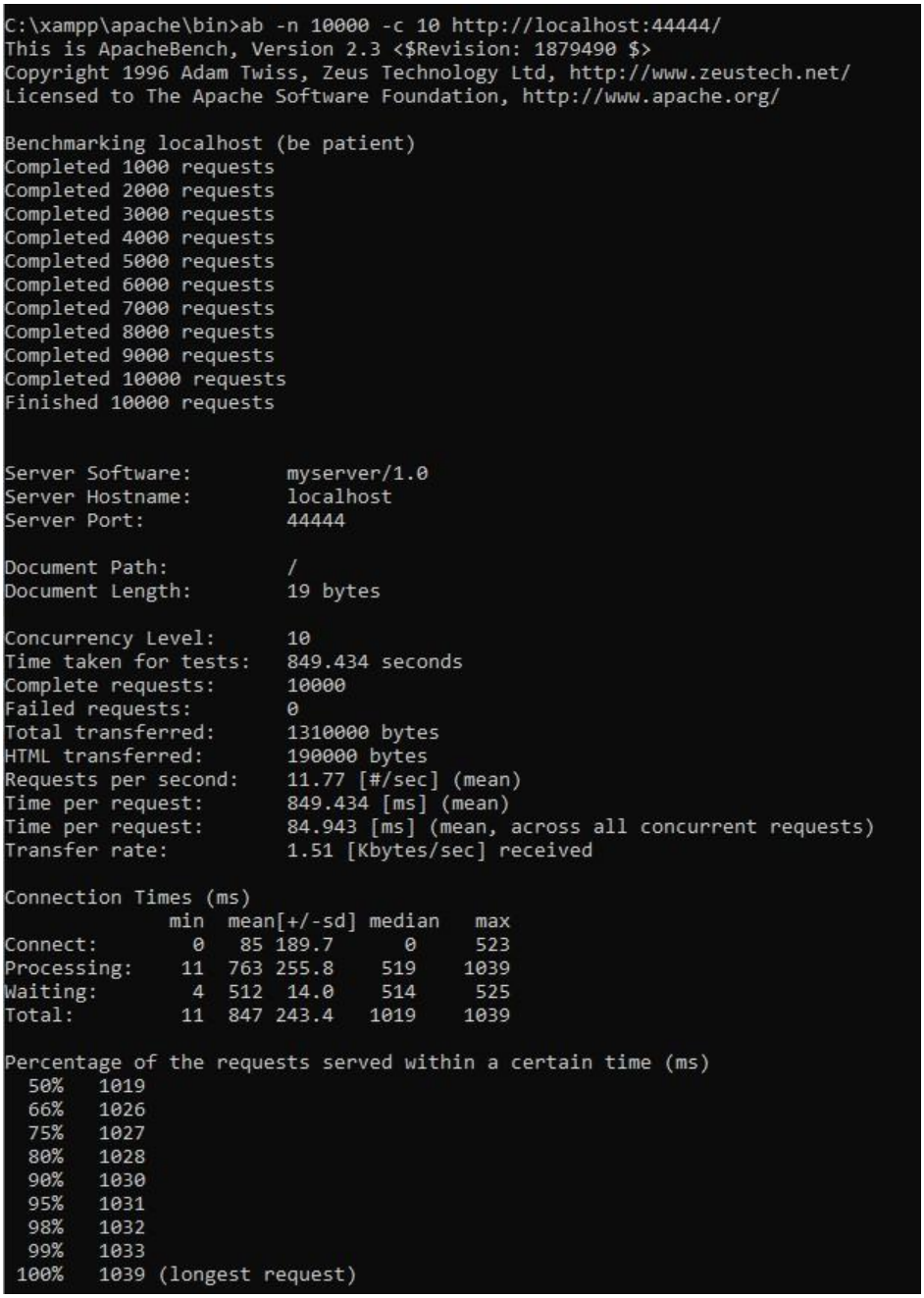
3. Membuka link url http://localhost:18000/pdf/rfc2616.pdf



Study Case 2

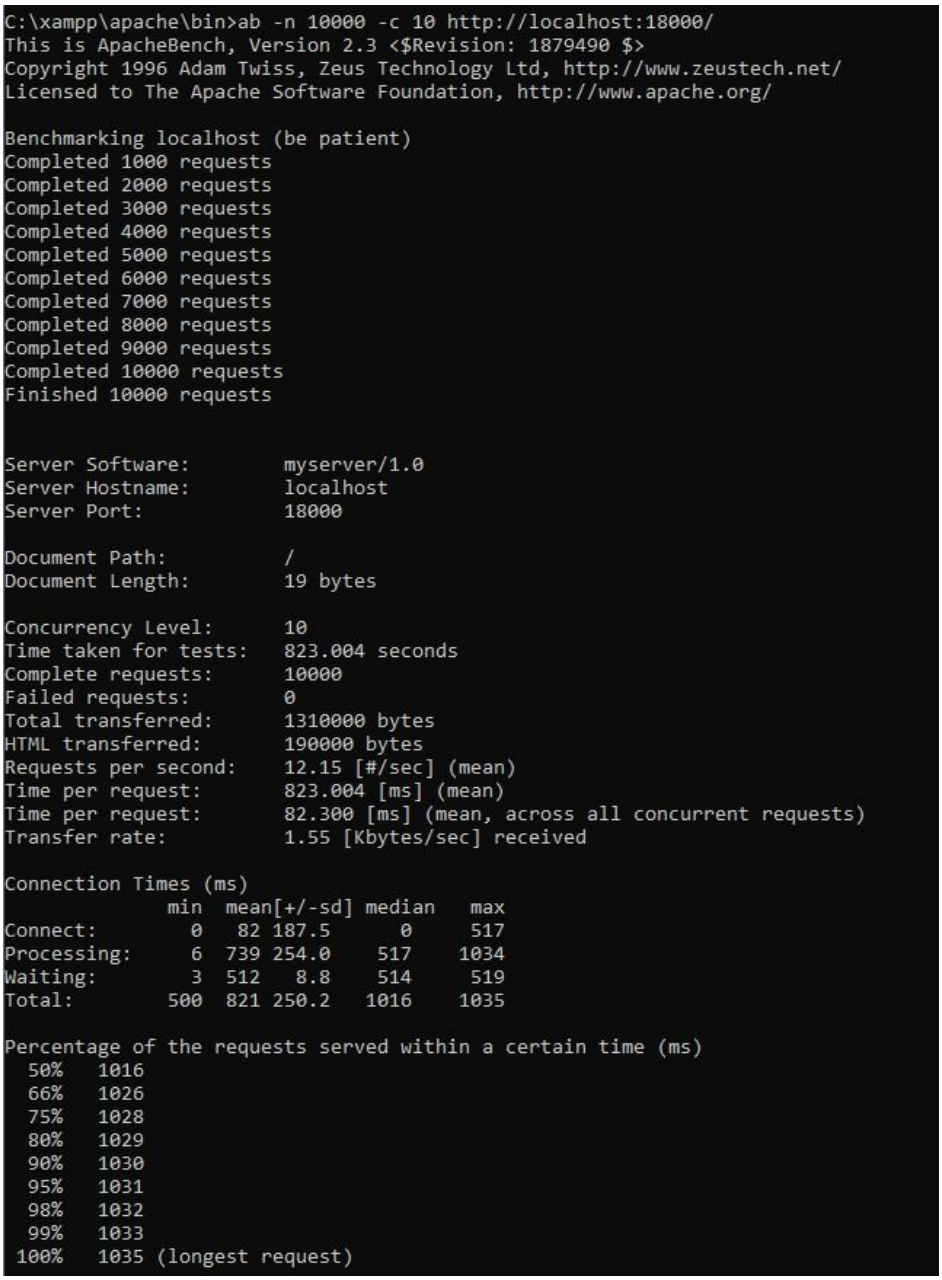
Berikut merupakan contoh screenshot testing apache untuk load balancing :

1. Load Balancing Async dengan 5 HTTP Server, dengan 10000 request dan 10 concurrency



```
C:\xampp\apache\bin>ab -n 10000 -c 10 http://localhost:44444/
This is ApacheBench, Version 2.3 <$Revision: 1879490 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking localhost (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:        myserver/1.0
Server Hostname:        localhost
Server Port:            44444

Document Path:          /
Document Length:        19 bytes

Concurrency Level:      10
Time taken for tests:   849.434 seconds
Complete requests:      10000
Failed requests:        0
Total transferred:      1310000 bytes
HTML transferred:       190000 bytes
Requests per second:    11.77 [#/sec] (mean)
Time per request:       849.434 [ms] (mean)
Time per request:       84.943 [ms] (mean, across all concurrent requests)
Transfer rate:          1.51 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0   85 189.7      0     523
Processing:    11  763 255.8    519    1039
Waiting:        4  512  14.0    514     525
Total:         11  847 243.4   1019    1039

Percentage of the requests served within a certain time (ms)
  50%   1019
  66%   1026
  75%   1027
  80%   1028
  90%   1030
  95%   1031
  98%   1032
  99%   1033
 100%   1039 (longest request)
```

2. Load Balancing Thread dengan 5 HTTP Server, dengan 10000 request dan 10 concurrency



```
C:\xampp\apache\bin>ab -n 10000 -c 10 http://localhost:18000/
This is ApacheBench, Version 2.3 <$Revision: 1879490 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking localhost (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:        myserver/1.0
Server Hostname:        localhost
Server Port:            18000

Document Path:          /
Document Length:        19 bytes

Concurrency Level:      10
Time taken for tests:   823.004 seconds
Complete requests:      10000
Failed requests:        0
Total transferred:      1310000 bytes
HTML transferred:       190000 bytes
Requests per second:    12.15 [#/sec] (mean)
Time per request:       823.004 [ms] (mean)
Time per request:       82.300 [ms] (mean, across all concurrent requests)
Transfer rate:          1.55 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0   82 187.5      0     517
Processing:     6  739 254.0    517    1034
Waiting:        3  512   8.8    514     519
Total:        500  821 250.2   1016    1035

Percentage of the requests served within a certain time (ms)
  50%   1016
  66%   1026
  75%   1028
  80%   1029
  90%   1030
  95%   1031
  98%   1032
  99%   1033
 100%   1035 (longest request)
```

Tabel APACHE

Load Balancing menggunakan Async

Jumlah HTTP Server	Concurrency	Jumlah Complete Request	Non-2xx response (ERROR)	Jumlah request per second	Time Per request (mean across) (ms)
1	2	10000	11	388.98	2.571
1	5	10000	622	877.76	1.139
1	10	10000	1451	16.1	60.208
2	2	10000	1	609.77	1.640
2	5	10000	0	680.93	1.469
2	10	10000	6	13.04	76.71
3	2	10000	3	477.98	2.092
3	5	10000	0	833.26	1.200
3	10	10000	0	12.09	82.724
4	2	10000	4	414.63	2.412
4	5	10000	0	759.18	1.317
4	10	10000	0	11.77	84.961
5	2	10000	0	335.88	2.977
5	5	10000	1	644.87	1.551
5	10	10000	0	11.77	84.943

Load Balancing menggunakan Thread

Jumlah HTTP Server	Concurrency	Jumlah Complete Request	Non-2xx response (ERROR)	Jumlah request per second	Time Per request (mean across) (ms)
1	2	10000	0	353.40	2.830
1	5	10000	0	561.87	1.780
1	10	10000	0	78.24	12.782
2	2	10000	0	372.22	2.687
2	5	10000	0	814.10	1.228
2	10	10000	0	12.22	81.844
3	2	10000	0	368.45	2.714
3	5	10000	0	760.66	1.315
3	10	10000	0	12.77	78.323
4	2	10000	0	288.36	3.468
4	5	10000	0	794.31	1.259
4	10	10000	0	12.33	81.109
5	2	10000	0	341.08	2.932
5	5	10000	0	829.91	1.205
5	10	10000	0	12.15	82.300

Kesimpulan

Menggunakan reverse proxy, server dapat dibagi menjadi tugas - tugas yang spesifik. Seperti server gambar, server pdf, server html, server video, dll. Semua terkoneksi menggunakan reverse proxy sehingga pengguna hanya perlu mengakses ip server proxy saja.
Selain itu, reverse proxy dapat berperan sebagai Load Balancing. Request dapat dibagi ke server - server sehingga tidak ada server yang mempunyai terlalu banyak request. Dengan reverse proxy, ip server satu per satu tadi juga terenkapsulasi, sehingga pengguna hanya perlu mengakses satu ip server saja dan requestnya akan langsung diarahkan ke server dengan load kecil.

Berdasarkan hasil percobaan apache, Load Balancing menggunakan async memungkinkan terjadinya failed request dibanding Load Balancing yang menggunakan threading. Selain itu semakin banyak jumlah http server juga mengurangi jumlah failed request pada Load Balancing yang menggunakan async.