JURNAL SISFOTENIKA
Vol. 10 No. 1, Januari 2020
p-ISSN: 2087-7897 ; e-ISSN : 2460-5344
DOI : 10.30700/jst.v10i1.773

Implementasi Load Balancing Web Server Menggunakan Apache di Ubuntu 16.04.

Implementation Load Balancing Web Server Using Apache on Ubuntu 16.04.

Kresno Wibowo*[1], Iskandar Fitri[2], Deny Hidayatullah[3]
[1,2,3]Universitas Nasional; Jl. Sawo Manila, RT.14/RW.3, Ps. Minggu, Kec. Ps. Minggu, Kota Jakarta Selatan, Daerah Khusus Ibukota Jakarta 12520, (021) 7806700
e-mail: *[1]kresnowibowo4598@gmail.com, [2]tektel2001@yahoo.com, [3]faradeari@gmail.com

Abstrak

Saat banyak pengguna yang mengakses suatu situs web pada saat yang bersamaan. Maka saat itu bisa membuat web server menampung beban yang berlebih karena tidak bisa menampung lagi banyaknya request yang diterima. Dan itu biasa terjadi saat web server tunggal atau single. Namun dengan menggunakan metode load balancing Round Robin dan juga Haproxy sebagai load balancer dapat membantu mengatur pembagian beban pada setiap server. Dengan pegujian yang dilakukan pada 250 user, 500 user, 750 user, sampai 1000 user dan menggunakan web statis. Hingga dapat melihat penggunaan CPU yang digunakan dengan beban user yang diuji pada saat setelah melakukan. Hasil dari pengujian dengan menggunakan haproxy berjalan dengan baik daripada single server dan dua server. Dengan menggunakan apache benchmark dalam menghitung Throughput dan Time per request antara single server, dua server serta menjalankan semua server menggunakan Haproxy. Untuk pengujian presentasi pada CPU yang dipakai pada 250 user yaitu 31,9% yang terendah namun terdapat peningkatan pada pengujian 1000 user yaitu mencapai 54,5%. Hasil pengujian pada Dua server dalam pengujian 250 user, 500 user, 750 user, dan 1000 user menghasilkan hasil 62.5%-68.7%. Dan dengan pengujian pada single server pemakaian RAM dalam pengujian 250 user, 500 user, 750 user, serta 1000 user yaitu mencapai sekitar 91%-96%. Dengan hasil pengujian yang lebih baik menggunakan load balancing haproxy daripada menggunakan single server maupun dua server. Karena penggunaan single server dan dua server hampir mencapai overload. Yang berarti waktu yang dibutuhkan untuk menangani beban tersebut ialah load balancing menggunakan haproxy lebih baik daripada single server untuk lebih cepat menangani request, time per second dan penggunaan CPU.

Kata kunci *— Apache, Haproxy, Load Balancing, Virtual Box*

Abstract

When many users access a website at the same time. So when it can make the web server to accommodate excessive loads because it can no longer accommodate the number of requests received. And that usually happens when a single web server or single. However, using the Round Robin load balancing method and also Haproxy as a load balancer can help regulate load sharing on each server. With testing done on 250 users, 500 users, 750 users, up to 1000 users and using a static web. So you can see the CPU usage that is used by the user load that is

tested at the time after performing. The results of testing using haproxy run better than a single server and two servers. By using Apache benchmark in calculating Throughput and Time per request between a single server, two servers and running all servers using Haproxy. For testing the presentation on the CPU that is used on 250 users is 31.9% the lowest but there is an increase in testing 1000 users, reaching 54.5%. The test results on two servers in testing 250 users, 500 users, 750 users, and 1000 users produced 62.5% -68.7% results. And by testing on a single server RAM usage in testing 250 users, 500 users, 750 users, and 1000 users which reached about 91% -96%. With better test results using haproxy load balancing than using a single server or two servers. Because the use of a single server and two servers almost reached overload. Which means the time needed to handle the load is load balancing using haproxy better than a single server to more quickly handle requests, time per second and CPU usage.

Keywords *— Apache, Haproxy, Load Balancing, Virtual Box*

1. PENDAHULUAN

Pada zaman sekarang yang sudah semakin maju era teknologi khususnya pada kemajuan bidang jaringan. Perkembangan tersebut terbilang cukup pesat setiap waktu. Meningkatnya kebutuhan akan informasi menuntut akses yang cepat untuk mendapatkan informasi – informasi terkini, salah satunya yang paling dominan mempengaruhi kecepatan akses suatu alamat website tertentu adalah server penyedia layanan. Server bisa dibilang sebagai pelayan karna tugasnya menerima request dari klien, permintaan datang dari klien yang sudah terhubung pada server yang ingin diakses, server merupakan pusat pengelolaan, semakin banyak klien semakin berat kerja yang dilakukan oleh server, sehingga dibutuhkan spesifikasi komputer khususnya server harus baik sehingga mampu melayani permintaan klien yang cukup banyak. Penelitian sebelumnya yang mengggunakan dua server untuk melakukan load balancing menggunakan nbl manager dilakukan oleh penulis pertama, dkk. Kemudian penelitian yang selanjutnya juga menggunakan metode nth yang dilakukan oleh penulis pertama, dkk [1].

Permasalahan pada umumnya tidak jarang pengelola server kurang mengelola dengan baik web server. Oleh sebab itu sering terjadi server down dan overload akibat banyaknya user yang mengakses secara bersamaan. Karena menggunakan satu server untuk menampung banyaknya permintaan. Karena menggunakan single server tidak ada server lain yang bisa membantu saat terjadi overload [2].

Load Balancing adalah suatu teknik untuk membagi beban pada koneksi secara seimbang, agar koneksi dapat berjalan lancar dan optimal, memaksimalkan throughput, memperkecil untuk terjadinya overload pada salah satu jalur koneksi Load balancing juga membagi beban kerja yang dilakukan secara merata di dua atau lebih komputer, link jaringan, CPU, hard drive, atau sumber daya lainnya, untuk mendapatkan pemanfaatan sumber daya yang optimal. dalam hal ini masalah pada pemerataan beban server menjadi salah satu solusi untuk dapat menjawab permasalahan yang ada diatas[3].

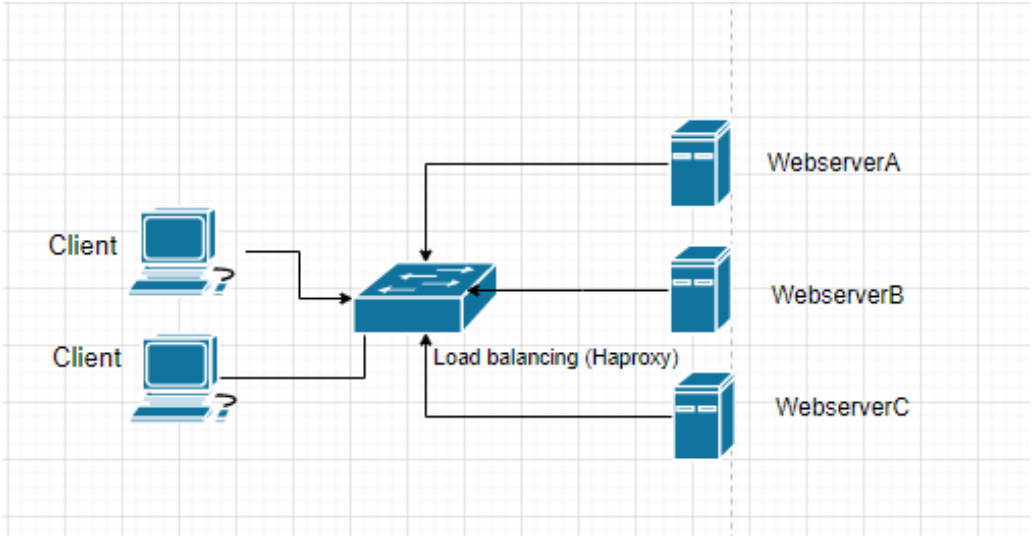
Penelitian ini bertujuan mengimplementasikan load balancing pada web server untuk menjamin kestabilan server dengan cara manajemen yang baik. Maanfaatnya Dapat memperkecil untuk terjadinya overload pada webserver. Dan dengan menggunakan load balancing dengan Algoritma Round Robin yang dapat membagi kinerja web server dengan merata. Karena dapat menjaga kestabilan server karena menggunakan lebih dari satu server.

2. METODE PENELITIAN

Perancangan yang akan dibuat ialah membangun sebuah *web server* yang akan nantinya dijadikan *load balancer* yaitu *haproxy*. Metode yang diterapkan dalam penelitian ini menggunakan *Load balancing dengan Algoritma Round Robin*. *Haproxy* akan mengambil *source* dengan beban yang setara antara satu *server* dengan *server* lainnya.

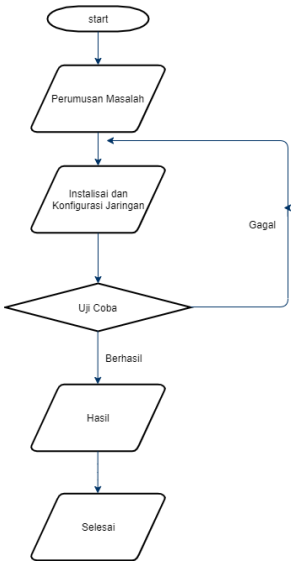
Apache adalah sebuah perangkat lunak *web server* yang menghubungkan antara *web* dengan *user*. *Apache* memudahkan pemilik *website* untuk membuat konten. *Apache* juga merupakan *software* lintas *platform* dan *server* nya dapat berjalan dengan baik.

Round Robin ialah sebuah *Algoritma* yang dipakai untuk mengambil *sourch* dengan beban yang setara antara *server* satu dengan *server* yang lainnya. Pembagian beban tersebut bertujuan agar saat banyak beban yang masuk secara bersamaan, kinerja server tidak down dan dapat berjalan dengan baik menerima beban.



Gambar 1. Topologi yang digunakan

Pada Gambar 1 menunjukkan jalan yang akan dilakukan pada sistem *load blancing* menggunakan 3 *server* dengan *OS ubuntu 16.04* yang masing-masing sudah diberi *web statis*. Lalu sebagai *load balancing* nya menggunakan *Haproxy* dengan *OS ubuntu 16.04* yang nantinya tersambung dengan *client*.



Gambar 2. Tahapan penilitan

Pada gambar 2 merupakan tahapan penelitian yang dilakukan mulai dari perumusan masalah lalu lanjut terhadap instalisasi yang dilakukan seperti install *Ubuntu 16.04* lalu *Apache* sebagai *web server* dan *Haproxy* sebagai *Load Balancer*. Setelah itu melakukan konfigurasi terhadap tiga server yang diuji dengan *haproxy*. Lalu lakukan pengujian namun jika pengujian gagal kita kembali melakukan konfigurasi. Sampai pada jika uji coba yang dilakukan berhasil masuk kepada tahap hasil yang diuji lalu selesai.

2. HASIL DAN PEMBAHASAN

A. Analisis Kebutuhan

Pada proses pembuatan aplikasi absensi dalam penelitian ini dibutuhkan beberapa perangkat pendukung seperti perangkat lunak (*software*) dan perangkat keras (*hardware*), yang diantaranya:

1. Kebutuhan Hardware : Laptop Intel Core i3-6006U, 2.0GHz dengan RAM 8GB dan HDD 1TB
2. Kebutuhan Software : Virtualbox, Google Chrome, Draw.io, Ms. Excel.

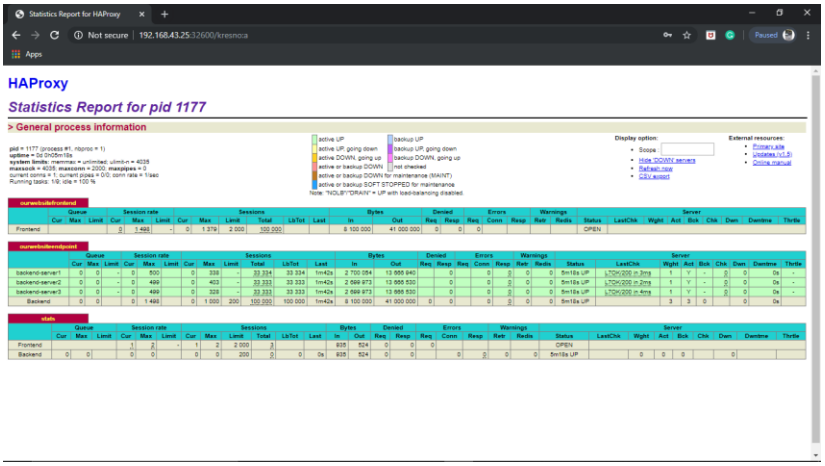
B. Tampilan Aplikasi

Pada proses ini tampilan web server dengan ip *address* yang digunakan untuk melakukan *Load Balancing*. *WebserverA* dengan *Ip Address* 192.168.43.10 sebagai server yaitu *Apache*. *WebserverB* dengan *Ip Address* 192.168.43.11 juga sebagai *server* yaitu *Apache*. *WebserverD* dengan *Ip Address* 192.168.43.13 juga sebagai *server* yaitu *Apache*. Lalu ada sebagai load balancer yaitu webserverL dengan Ip Adress 192.168.43.25 dengan *load balancer Haproxy* yang digunakan.

Tabel 1. IP *Adress*

Nama	Ip Address	Isi
webserverA	192.168.43.10	Apache
webserverB	192.168.43.11	Apache
webserverD	192.168.43.13	Apache
webserverL	192.168.43.25	Haproxy

Haproxy Stats ialah hasil yang akan ditampilkan saat sudah melakukan *load balancing web server*. Untuk mengetahui hasil tersebut maka terlebih dahulu melakukan *testing* terhadap *load balancer*.



Gambar 2. Statistik tampilan Haproxy

Gambar 2 Berikut adalah hasil yang ditampilkan dari hasil static yang nanti tiga web server tersebut akan dilakukan *load balancing* dengan memasukkan data yang akan diuji dengan diberi beban yang ingin diuji.

ourwebsiteendpoint

	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
backend-server1	0	0	-	0	500		0	338	-	33 334	33 334	1m42s	2 700 054	13 666 940		0		0	0	0	0	5m18s UP	L7OK/200 in 3ms	1	Y	-	0	0	0s	-
backend-server2	0	0	-	0	499		0	403	-	33 333	33 333	1m42s	2 699 973	13 666 530		0		0	0	0	0	5m18s UP	L7OK/200 in 2ms	1	Y	-	0	0	0s	-
backend-server3	0	0	-	0	499		0	328	-	33 333	33 333	1m42s	2 699 973	13 666 530		0		0	0	0	0	5m18s UP	L7OK/200 in 4ms	1	Y	-	0	0	0s	-
Backend	0	0		0	1 498		0	1 000	200	100 000	100 000	1m42s	8 100 000	41 000 000	0	0		0	0	0	0	5m18s UP		3	3	0		0	0s	

Gambar 3. Hasil yang ditampilkan

Pada Gambar 3. Seperti yang dilihat pada gambar 3 ialah tampilan lebih jelas yang akan keluar setelah melakukan *load balancing* dengan menggunakan load balancer haproxy yaitu dapat menampilkan tiga server hasil dari pengujian.

C. Pengujian

Dalam proses pengujian yang akan dilakukan pada penelitian ini ialah memasukkan dari 250, 500, 750 sampai 1000 permintaan akses *user*. Pengujian tersebut meliputi hasil pengujian *request persecond, transfer rate*, dan penggunaan memory.

```
frontend ourwebsitefrontend
        bind *:80
        mode http
        default_backend ourwebsiteendpoint

backend ourwebsiteendpoint
    balance roundrobin
    option forwardfor
    http-request set-header X-Forwarded-Port [dst_port]
    http-request set-header X-Forwarded-Proto https if { ssl_fc }
    option httpchk HEAD / HTTP/1.1\r\nHost:localhost
    server backend-server1 192.168.43.11:8080 check
    server backend-server2 192.168.43.10:8080 check
    server backend-server3 192.168.43.25:8080 check

listen stats
    bind :32600
    stats enable
    stats uri /
    stats hide-version
    stats auth kresno:a
```

Gambar 4. Konfigurasi Haproxy

Pada gambar 4. Terlihat pada gambar 4 yaitu saat melakukan konfigurasi untuk menghubungkung antara *web server* 1, *web server* 2, *web server* 3 dengan *load balancer*. Untuk melakukan *load balancing*, agar masuk ke *konfigurasi* pada Gambar 4 *nano /etc/haproxy/haproxy.cfg*, harus memasukkan *ip address web server* yang akan digunakan seperti Gambar 4 untuk *web server* 1 -192.168.43.10, *web server* 2 -192.168.43.11 *web server* 3 -192.168.43.13 dan untuk haproxy sendiri memiliki ip *address* yaitu web serverL - 192.168.43.25.

```
root@ubuntu:/home/kresno# ab -n 10000 -c 1000 http://192.168.43.25/
```

Gambar 5. Apache benchmark

Pada gambar 5, terlihat pada gambar 5 ialah Untuk pengujian menggunakan *Apache Benchmark* memasukkan script seperti Gambar 5. Parameter c adalah untuk menguji jumlah *request* yang ingin diuji, sedangkan parameter n adalah jumlah koneksi yang nantinya dibuat untuk ke server tujuan, lalu masukkan ip *addres* tujuan yang ingin diuji yaitu *load balancer* nya.

```
Server Software:        Apache/2.4.18
Server Hostname:        192.168.43.25
Server Port:            80

Document Path:          /
Document Length:        140 bytes

Concurrency Level:      1000
Time taken for tests:   14.219 seconds
Complete requests:      10000
Failed requests:        0
Total transferred:      4100000 bytes
HTML transferred:       1400000 bytes
Requests per second:    703.27 [#/sec] (mean)
Time per request:       1421.937 [ms] (mean)
Time per request:       1.422 [ms] (mean, across all concurrent requests)
Transfer rate:          281.58 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0   10  39.8      1    1000
Processing:    47  898 869.8    714   14097
Waiting:       40  896 869.9    713   14097
Total:        143  908 875.3    716   14164
```

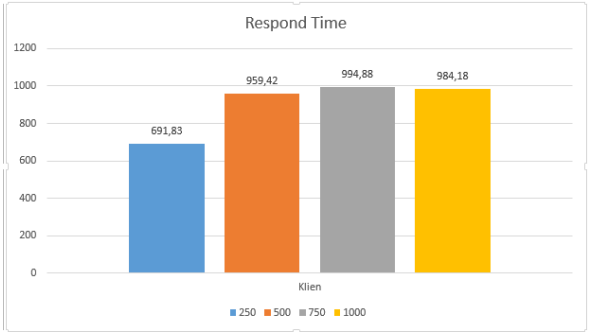
Gambar 6. Hasil Apache benchmark

Pada gamber 6, terlihat dari hasil pengujian dengan 1000 user ialah untuk hasil dari time per request menghasilkan angka 1.033, dan untuk transfer rate yaitu dengan hasil 387.49 (kbytes/sec),begitu juga dengan hasil dari *time takes for tests* menghasilkan 10.333 *second*, dan juga hasil dari *request per second* yaitu mencapai 967.77 (/sec).

Tabel 2. Pengujian kinerja RAM

Jumlah User	Nama Server	%of CPU	Memory
250	webserverA	31,9%	227
	webserverB	33,6%	225
	webserverD	34,3%	226
500	webserverA	42,6%	220
	webserverB	42,3%	256
	webserverD	47,2%	252
750	webserverA	39,1%	275
	webserverB	43,5%	274
	webserverD	49,3%	273
1000	webserverA	38,7%	256
	webserverB	45,8%	274
	webserverD	54,4%	277

Pada Tabel 2 adalah hasil penggunaan RAM dengan menggunakan htop setelah melakukan pengujian. Dari hasil pengujian % of RAM yang telah diringkas menjadi tabel agar mudah membacanya. Pembagian pada beban 3 *web server ubuntu* yang tersedia menggunakan *Haproxy* sebagai *load balancer* secara fungsional. Pada pengujian 250 user untuk pembagian beban yang hampir setara yaitu dengan hasil diangka 31-34%. Di 500 user juga hasil pembagian hampir setara yaitu dihasil 42-47%. Di 750 *user* mulai terjadi perbedaan karena web serverA menghasilkan 39,1% sementara web serverB 43,5% dan web serverD menghasilkan 49,3%. Dari web serverA dan D memiliki selesih 10%. Di 1000 user web serverA.



Gambar 6. Respond Time

Pada gambar 6, terlihat gambar 6 menunjukkan hasil dari saat melakukan *Apache Bencmark* untuk *request per second* yang pada pengujian 250 *user* yang berada pada angka 691.83 *per second*. untuk pengujian pada 500 *user* yaitu menghasilkan pada angka 959.43 *per second*. Pada pengujian untuk 750 *user* yaitu mencapai angka 994.88 *per second*. dan untuk pengujian terakhir yaitu pada 1000 *user* menghasilkan angka mencapai 984.18 *per second*.

Tabel 3. Pengujian single server

Nama	User	%ofCPU	Memori	Request per second
Webserver	250	91,9%	225	1061.31
Webserver	500	93,4%	229	1004.27
Webserver	750	95,5%	231	1060.485
Webserver	1000	96.0%	235	1313.786

Pada Tabel 3 menujukkan kinerja *server* yang sangat besar jika menggunakan *single server*. Karena tidak ada yang bisa membatu saat banyak user mengakses dengan jumlah besar. Oleh karena itu akhirnya sudah hampir memenuhi kapasitas dan terjadi *overload*. Terlihat penggunaan yang digunakan pada pengujian 250 *user* yaitu mencapai angka 91,9%, dan pengujian 500 *user* juga mencapai hasil 93,4%, sementara untuk pengujian 750 *user* mencapai angka 95,5%, dan untuk pengujian 1000 *user* yaitu mencapai angka tertinggi yaitu 96,0%. Ini adalah hasil jika penggunaan single server saat melalukan menerima beban yang cukup besar. Angka yang dihasilkan cukup tinggi dikarenakan server hanya bekerja sendiri tanpa ada pembagian beban ke server lainnya.

Tablel 4. Pengujian dua server

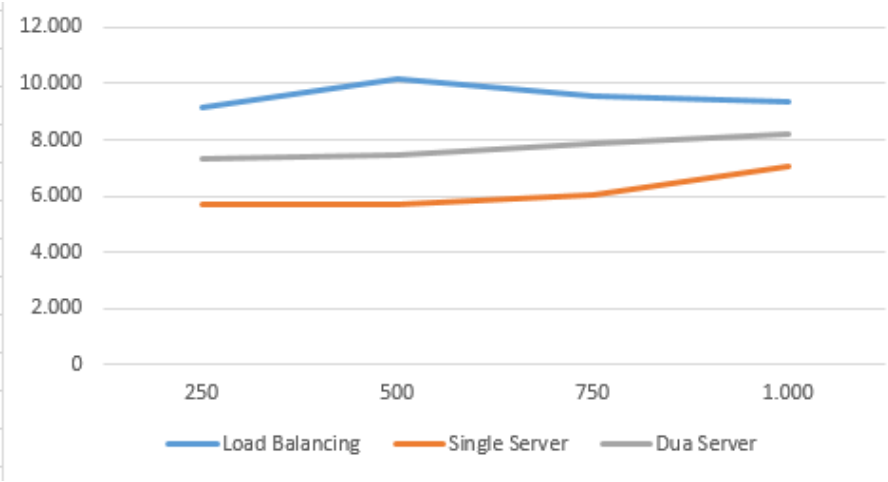
User	Nama	%ofCPU	Memori	Request per Second
250	webserverA	65,3%	223	1134.44
	webserverB	67.8%	225	
500	webserverA	63.4%	231	1087.50
	webserverB	66.2%	236	
750	webserverA	67,8%	232	1091.93
	webserverB	68.4%	229	
1000	webserverA	68.7%	237	1023.30
	webserverB	62.5%	235	

Pada tabel 4. Terlihat pengujian menggunakan *dua server* atau *satu server* yang dimatikan terlihat bahwa penggunaan RAM cukup meningkat jika tidak menggunakan semua *server*. Yaitu dengan hasil pengujian dari 250 *user* untuk *webserverA* mencapai 65,3% sementara untuk *webserverB* mencapai 67,8%. Untuk hasil pengujian dari 500 *user* untuk *webserverA* sendiri mencapai 63,4% sementara untuk *webserverB* mencapai 66,2%. Dan untuk hasil dari pengujian dari 750 *user* juga untuk *webserverA* mencapai 67,8% sedangkan untuk webserverB menghasilkan 68,4%. Dan untuk pengujian terakhir yaitu 1000 *user* untuk *webserverA* mencapai 68,7% dan hasil lebih rendah yaitu *webserverB* dengan hasil 62,5%. Meskipun menggunakan *dua server* tetap penggunaannya masih diatas 50% atau setengah lebih dari kapasitas seharusnya.

Tabel 5. Pengujian hasil load balancing

User	Time taken for test [second]	Request per second [#/sec]	Time per request [ms]	Transfer rate [kbytes/sec]
250	9.029	1083.40	0.923	433.78
500	9.060	1103.80	0.906	441.95
750	9.244	1081.76	0.924	433.13
1000	9.284	1077.16	0.928	431.28

Pada tabel 5 Menampilkan hasil dari *time taken for test, request per second, time per request,* dan *juga transfer rate*. Karena menggunakan *Algoritma Round Robin* maka bisa dilihat pembagian yang merata pada pengujian *load balancing*. Hasil pengujian pada tabel 5 dari *time taken for test* untuk pengujian 250 *user* ialah 9.029, 500 user ialah 9.060, 750 user ialah 9.244, dan untuk 1000 user mencapai 9.284. Dan hasil *request per second* dari 250 *user* ialah 1083.40/sec, 500 *user* mencapai 1103.80/sec, dan 750 *user* dengan hasil 1081.76/sec, dan untuk 1000 *user* mencapai 1077.16. untuk *time per request* hasil dari 250 *user* ialah 0.923, 500 *user* mencapai 0.906, 750 *user* mencapai angka 0.924, lalu untuk 1000 *user* mencapai angka 0.928. dan untuk hasil dari *transfer rate* dari hasil 250 user mencapai 433.78, untuk 500 user mencapai 441.95, dan 750 user mencapai 433.13, dan yang terakhir untuk 1000 user mencapai 431.28. Hasil tersebut cukup stabil saat melakukan load balancing karena pembagian yang cukup merata.

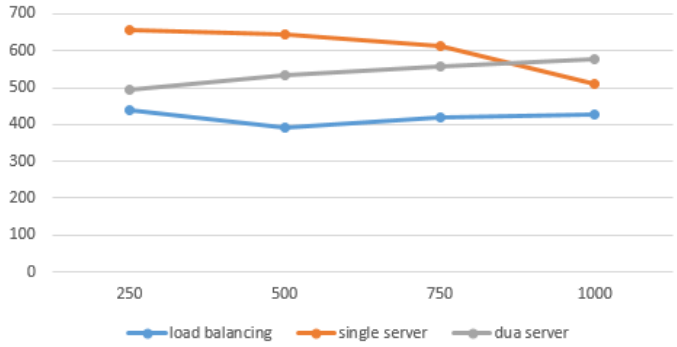


Gambar 7. Perbandingan Single Server dengan Load Balancing

Berdasarkan pada pengujian gambar 7, data yang telah dihasilkan dari waktu *respon* ialah diketahui dari 500 *client, single server* dapat melayani dengan baik setiap data yang masuk. Namun ketika diatas 500 client maka kinerja nya mulai berkurang akibat banyaknya data yang masuk. Begitupun dengan dua server tidak berbeda jauh dengan single server, karena pembagian bebannya hanya ke satu server tersisa walaupun tetap bisa membagi beban namun dapat berjalan dengan baik. Namun tidak dengan menggunakan *load balancing* karena ada pembagian beban yang dilakukan saat melayani jumlah user yang meningkat.

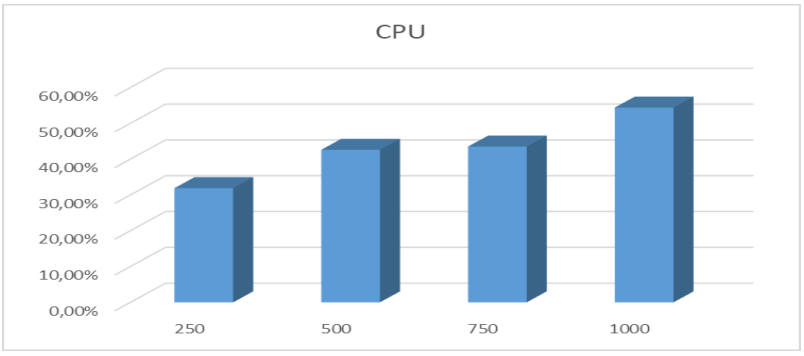
Gambar 8. Perbandingan *request*

Pada gambar 8 hasil dari pengujian *request.* Terlihat bahwa menggunakan *load balancing* sangat berjalan dengan baik dibandingkan dengan single server maupun dua server. Dengan menggunakan single server terlihat setiap request yang masuk maka sudah akan terjadinya overload dari server tersebut karena server hanya bekerja sendiri disaat masuknya request. Seperti halnya single server, dua server juga tidak berbeda jauh meskipun ada satu server lagi yang membantu disaat masuknya request. Dan menggunakan load balancing tiga server baru mulai dapat membagi beban dengan stabil dan berjalan cukup baik.



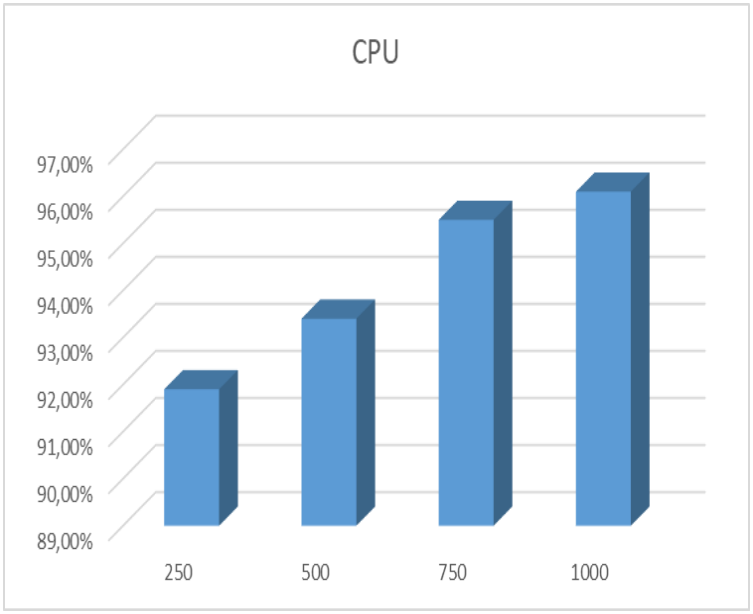
Gambar 9. Perbandingan Transfer Rate

Pada pengujian gambar 9. *Transfer Rate* yang menggunakan *load balancing* lebih stabil dan lebih kecil dibandingkan dengan *single server* yang cukup tinggi dan perbedaannya cukup terlihat. Sedangkan *dua server* berada diantara *single server* dan *load balancing*. Pada *load balancing* berada diangka 393.83-438.87[kbytes/sec]. dan untuk *dua server* 496.73-578.65[kbytes/sec]. sedangkan untuk pengujian *single server* yaitu 512.03-656.49[kbytes/sec].



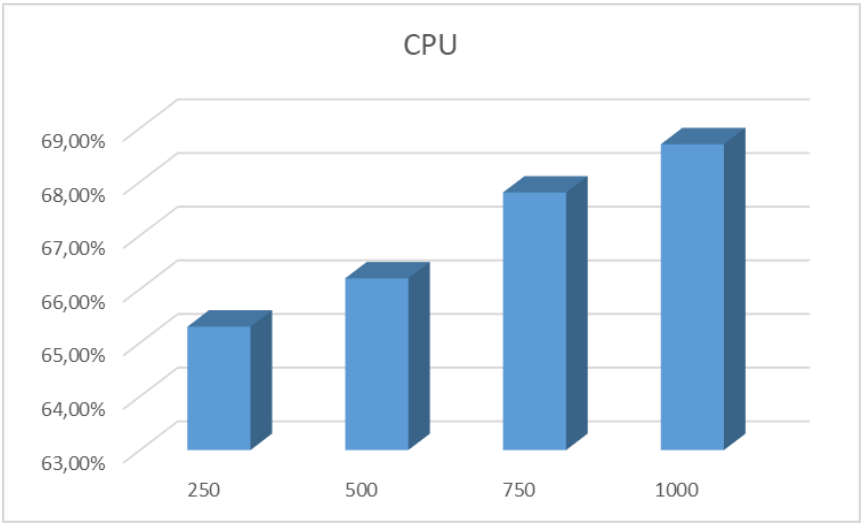
Gambar 10. Penggunaan Memori load balancing

Pada pengujian gambar 10. Penggunaan Memori pada pengujian load balancing terlihat lebih stabil dan tidak terlalu berbeda jauh. Oleh karena itu penggunaannya berjalan dengan baik dengan pengujian mulai dari 250 user yaitu 32,9%, 500 user mencapai 42.6%, pada 750 user berada pada angka 43.5%, dan untuk 1000 user 54.4%. setiap penggunaan yang lebih besar, maka terdapat peningkatan pemakaian memori pada setiap prosesnya.



Gambar 11. Penggunaan memori single server

Pada pengujian gambar 11. Pengujian pada *single server* terlihat pada pengujian tersebut bahwa penggunaan single server tidak berjalan dengan baik karena memiliki kapasitas yang tidak cukup besar. Seperti yang sudah diuji dengan 250 *user* yaitu berada pada angka 91,9%, sedangkan 500 *user* 93,4%, lalu 750 *user* mencapai 95.5%, dan untuk 1000 *user* mencapai 96.1%. terlihat pada pengujian menggunakan *single server* kapasitas yang digunakan hampir *overload*. Karena saat melakukan proses tidak ada yang dapat membantu saat banyaknya data yang masuk. Karena *single server* bekerja sendiri. Ini membuktikan bahwa *single server* tidak cukup untuk menampung data yang cukup banyak.



Gambar 12. Penggunaan memori dua server

Pada pengujian gambar 12. Terlihat pengujian pada *dua server* dapat membagi beban yang ditanggung walaupun masih di angka yang cukup tinggi yaitu masih diatas 50% yaitu yang berarti melebihi dari setengah kapasitas yang dapat digunakan. Seperti pengujian yang sudah dilakukan dengan pengujian mulai dari 250 *user* mendapatkan hasil 65,3%, 500 user mencapai 66,2%, dan 750 mencapai 67,8%, dan 1000 user mencapai 68,7%. Terlihat pada pengujian dua server pembagian beban cukup berjalan dengan baik walaupun hanya menggunakan *dua server*. Karena saat data masuk *ser*ver tidak bekerja sendirian seperti halnya saat menggunakan *single server*.

4. KESIMPULAN

Dari hasil uji coba dalam pembahasan di atas, maka dapat disimpulkan:

a. Secara fungsi *load balancing* menggunakan *Haproxy* berjalan dengan baik. Dengan menggunakan *tiga web server* dengan *Apache.*
b. Untuk pengujian presentasi pada CPU yang dipakai pada 250 user yaitu 31,9% yang terendah namun terdapat peningkatan pada pengujian 1000 user yaitu mencapai 54,5%. Namun pada titik tertinggi itu *load balancing* menggunakan *Haproxy* berjalan dengan baik pada pembagian beban kepada 3 server tersebut. Dan dapat memperlihatkan bila semakin besar beban user yang mengakses, maka semakin besar juga kinerja *web* untuk bekerja.
c. Untuk hasil pengujian menggunakan *single server* cukup terlihat beban yang besar terjadi. Terlihat pemakaian RAM dalam pengujian 250, 500, 750, 1000 user yaitu sekitar 91%-96% ini bisa terjadi *overload* karena penggunaan yang hampir 100% ketika dijalankan. Dan penggunaan *dua server* berada di sekitar 62.5%-68.7% pada hasil dari pengujian.

5. SARAN

Dari hasil penelitian yang telah dilakukan dengan menggunakan tiga server dengan satu load balancer cukup berjalan dengan baik. Yang data yang di uji muai dari 250, 500, 750 sampai 1000 user. Dengan hasil pada pengujian CPU yang dipakai pada pengujian 250 user yaitu 31,9% yang terendah namun mendapatkan peningkatan pada pengujian 1000 user 54,5%. Namun pada penggunaan single server terlihat kapasitas yang hampir overload karena server hanya bekerja

sendiri. Pada pengujian dua server cukup baik pembagian beban walaupun penggunaan juga masih diatas 50%. Namun saat dicoba melebihi 1000 mulai tidak dapat menampung banyaknya data yang diterima. Oleh karena itu disarankan jika ingin menggunakan kapasitas lebih besar disarankan untuk menambah lagi beberapa server agar dapat menampung data yang lebih besar lagi.

DAFTAR PUSTAKA

[1]. Muhammdad Sholikin, Muhammad Akbar, Zaid Amin "Analisis Performasi Server Cluster Pada Load Balancing Web Server Menggunakan NBL MANAGER".

[2]. Denny Rachmawan, Dadan Irwan, Harum Argyawati, 2016, "Penerapan Teknik Load Balancing pada Web Server lokar dengan metode Nth menggunakan Mikrotik"

[3]. Alam Rahmatulloh, Firmansyah MSN 2017, "Implementasi load balancing web server menggunakan haproxy dan sinkronisasi file pada sistem informasi akademik Universitas Siliwangi".

[4]. Syaqia Azizah, Asmunin, 2017, "Implementasi Load Balancing web server menggunakan Haproxy", Jurnal Management Informatika, Vol 8 ,No.01, 2017.

[5]. Efendi Yusuf, Tengku A Riza, Tody Ariefianto, 2013, "Impelemtasi Teknologi Load Balancer denga web server nginx untuk mengatasi beban server", STMIK AMIKOM Yogyakarta, Januari 2013.

[6]. Fajar Zuhroni, Adian Fatchur Rochim, Eko didik Wudianto, "Analisa Performasi Layanan Kluster Server Menggunakan Penyeimbang Beban dan Virtualbox". Jurnal Teknologi dan Sistem Komputer, Vol 3, No.4, Oktober 2015

[7]. Abe Wisnu Syaputra, Setiawan Asseegaff 2017, "Analisis dan Implementasi Load Balancing dengan Metode NTH pada Jaringan Dinas Pendidikan Provinsi Jambi," 2017 JurnalManajemenSistemInformasi Vol.2, No.4, Desember 2017

[8]. Sampurna Dadi Riskiono, 2018, "Implementasi Metode Load Balancing dalam mendukung Sistem Kluster Server".

[9]. Syamsul Alam Haris, Hero Suhartono, Herlawati, 2018, "Menjaga Kestabilan Jaringan Load Balancing Nth dengan teknik Failover Pada PT. Jakarta Samudera Sentosa Jakarta".

[10]. Warman I, Andrian A. 2017. Analisis Kinerja Load Balancing Dua line Koneksi Dengan Metode Nth. Jurnal TEKNOIF 5(1): 56-62

[11]. Futri Utami, Lindawati Lindawati, Suzan Zefi, 2017, "Optimalisasi load balancing dua ISP untuk manajemen bandwidth berbasis Mikrotik".