Implementasi Sistem Load Balancing Web server Pada Jaringan Public Cloud Computing Menggunakan Least Connection

Aina Fadila[1], Muhammad Nasir[2], Safriadi[3*]

[1,3] *Jurusan Tekniknologi Informasi dan Komputer Politeknik Negeri Lhokseumawe*
Jln. B.Aceh Medan Km.280 Buketrata 24301 INDONESIA

[1]ainafadilaoktober22@gmail.com

[2]muhnasir.tmj@pnl.ac.id

[3*]safriadi@pnl.ac.id

Abstrak— ***Web* adalah sebuah perangkat lunak dengan berbasis data yang berfungsi untuk menerima permintaan dari *client* dan tanggapan permintaan dengan mentranfer melalui browser yang merupakan halaman situs *web*. Dibalik kemudahan pengaksesan segala informasi terdapat permasalahan yang terjadi pada trafik yang menuju web server yaitu dengan meningkatnya permintaan informasi akan dapat menjadikan trafik menuju web server menjadi *overload* dan akhirnya menjadi *down* karena tidak mampu menjalankan permintaan yang berlebihan. Untuk mengatasi permasalahan tersebut adalah dengan menggunakan load balancing yang bertugas untuk mendistribusikan beban trafik kebanyak server. Rumusan masalah yang terdapat adalah Bagaimana sitem monitoring jalanya trafik secara *real time* dan Bagaimana performa web server yang menggunakan *load balancing* dan *web server* tidak menggunakan *load balancing*. Tujuannya untuk melihat system monitoring secara real time dan mengetahui performa *web server* menggunakan *load balancing* dan tidak menggunakan *load balancing*.Pada penelitian ini diselesaikan dengan menerapkan *load balancing* pada jaringan public dan menerapkan *load balancing Haproxy* pada server serta didukung algoritma *least connetion.* Bedasarkan analisis, dapat diperoleh hasil bahwa keberhasilan s*ystem* jalannya *traffic* secara real time yaitu 90 % dan hasil uji performa dari *web server* menggunakan aplikasi jmeter dengan jumlah *traffic* 1000 permintaan dalam satu waktu dengan looping 1,10,50 dan 100 pada load balancing nilai rata-rata *throughput* 630.2/sec dan tidak menggunakan load balancing nilai rata-rata *throughput* 354.5/sec.**

Kata kunci : *Load balancing, Web Server*, Apache JMeter, *Docker*

Abstract— **Web is a software with data-based that functions to receive requests from clients and respond to requests by transferring through a browser which is a website page. Behind the ease of accessing all information, there are problems that occur in traffic to the web server, namely with the increase in requests for information, it will be able to make traffic to the web server become overloaded and eventually down because it is unable to carry out excessive requests. To overcome this problem is to use load balancing which is in charge of distributing traffic loads to many servers. The formulation of the problem is how the system monitors traffic in real time and how the performance of web servers that use load balancing and web servers do not use load balancing. The goal is to see the monitoring system in real time and find out the performance of the web server using load balancing and not using load balancing. This research was completed by applying load balancing on public networks and applying Haproxy load balancing on servers and supported by least connetion algorithms. Based on the analysis, and the results of performance tests from the web server using the JMet application with the number of traffic 1000 requests at one time with looping 1, 10, 50 and 100 on load balancing average throughput value of 164.2 / sec and not using load balancing average throughput value of 612.2 / sec.**

Keywords— Load balancing, Web Server, Apache JMeter, Docker.

I. PENDAHULUAN

Perkembangan teknologi informasi yang semakin pesat memasuki era modern ini layanan *web* semakin meningkat setiap saat. dapat memudahkan pengguna internet dalam mengakses informasi yang dibutuhkan dalam waktu yang cepat melalui sebuah *web*.Salah satu jenis *web* adalah sebuah perangkat lunak dengan berbasis data yang berfungsi untuk menerima permintaan dari *client* dan tanggapan permintaan dengan mentranfer melalui browser yang merupakan halaman situs *web*.[1].

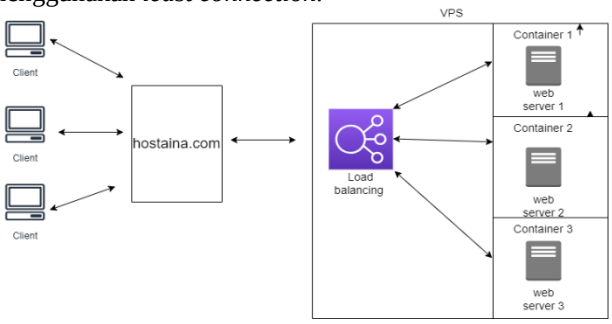
Dibalik kemudahan pengaksesan segala informasi terdapat permasalahan yang terjadi pada trafik yang menuju web server yaitu dengan meningkatnya permintaan informasi akan dapat menjadikan trafik menuju web server menjadi *overload* dan akhirnya menjadi *down* karena tidak mampu menjalankan permintaan yang berlebihan.[2] Untuk mengatasi permasalahan tersebut adalah dengan menggunakan load balancing yang bertugas untuk mendistribusikan beban trafik kebanyak server.[3]

Penelitian ini diselesaikan dengan judul implementasi *system Load balancing web server* pada jaringan public *cloud computing* menggunakan *Least Connection*" pada penelitian ini adalah jaringan *public* yang dapat di akses oleh semua pengguna. Menerapkan *load balancing* menggunakan nama domain dan hanya menggunakan 1 komputer yang digunakan sebagai *server* yang didalamnya terdapat 3 *web server* dengan menerapakan *load balancing haproxy* yang didukung algoritma *Least Connetion* . Pada metode *least connection* ini membagi beban berdasarkan banyaknya koneksi yang sedang dilayani oleh *server*. *Server* yang memiliki koneksi paling sedikit akan melayani permintaan yang masuk.

II. METODOLOGI PENELITIAN

A. Blok Diagram Sistem

Berikut merupakan Blok diagram system yang digunakan dalam proses implementasi sostem *load balancing web server* pada jaringan *public cloud computing* menggunakan *least connection.*



Gambar 1 Blok Diagram Sistem

Berdasarkan gambar 1 Secara singkat penjelasan mengenai cara kerja dari sistem ini yaitu dimulai dari *client* yang mengirim permintaan akses ke situs *web server* dengan melalui host.aina22.com. Permintaan dari *client* akan masuk ke *Load balancing* terlebih dahulu, kemu dian dari *Load balancing* permintaan *client* akan diteruskan ke beberapa *web server* yang kemudian akan dilakukan pembagian beban terhadap beberapa *web server*. Maka *web server* akan memproses permintaan *client.* Selanjutnya akan dilakukan pengamatan pembagian beban *Load balancing* ke *web server* sudah bekerja dengan baik atau tidak.

B. Metode Penelitian

Metode yang akan digunakan dalam penelitian ini menggunakan Metode *least connetion*. Dalam rancangan sistemnya penelitian ini memuat perancangan blok diagram sistem secara keseluruhan yang menjelaskan bagaimana penerapan metode *least connetion* Algoritma ini membagi beban berdasarkan banyaknya koneksi yang sedang dilayani oleh *server*. *Server* yang memiliki koneksi paling sedikit akan melayani permintaan yang masuk.

C. Instal Haproxy

Haproxy digunakan untuk mendistribusikan atau menyeimbangkan Trafik paket data dari debian ke*server*.

```
root@host:~# apt-get install haproxy
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
haproxy is already the newest version (2.2.9-2+deb11u5).
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
```

Gambar 2 Install Haproxy

Gambar 2 adalah *HAProxy* bertujuan untuk menerapakn *load balancing* yang andal, *fleksibel*, dan kuat yang telah digunakan secara luas oleh banyak organisasi dan situs web besar. Penggunaan *HAProxy* sebagai *load balancer* dapat membantu meningkatkan kinerja, ketersediaan, dan skalabilitas aplikasi.

D. Konfigurasi Haproxy

Pada bagian konfigurasi *haproxy* diterapkan algoritma Least connection

```
frontend http_front
                log global
                bind *:80
                mode http
                default_backend backend_server

backend backend_server
        mode http
        balance leastconn
        option forwardfor
        server webserver_1 localhost:81 check
        server webserver_2 localhost:82 check
        server webserver_3 localhost:83 check

listen stats
       bind *:8080
       stats enable
       stats hide-version
       stats refresh 7s
       stats show-node
       stats auth root:ainafadila22
       stats uri /haproxy
```

Gambar 3 Konfigurasi Haproxy

Gambar 3 pada bagian frontend http_front akses awalnya web serve. Pada bagian backend_server menggunakan tiga web server dimana web server 1, web server2 dan web server 3. Kemudian bagian listen stats untuk mengaktifkan *haproxy* atau mengatur url untuk membuka halaman dashboard *haproxy* melalui browser.

E. Konfigurasi Docker Compose

```
root@host:~# docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED       STATUS       PORTS                                   NAMES
32d075a0b2e5   httpd     "bash -c 'echo 'Dire…"   20 hours ago  Up 20 hours  0.0.0.0:81->80/tcp, :::81->80/tcp       websi
te-webserver1-1
d2f23bcd0650   httpd     "bash -c 'echo 'Dire…"   20 hours ago  Up 20 hours  0.0.0.0:82->80/tcp, :::82->80/tcp       websi
te-webserver2-1
0d49e3063459   httpd     "bash -c 'echo 'Dire…"   20 hours ago  Up 20 hours  0.0.0.0:83->80/tcp, :::83->80/tcp       websi
te-webserver3-1
```

Gambar 4 Konfigurasi *web server*

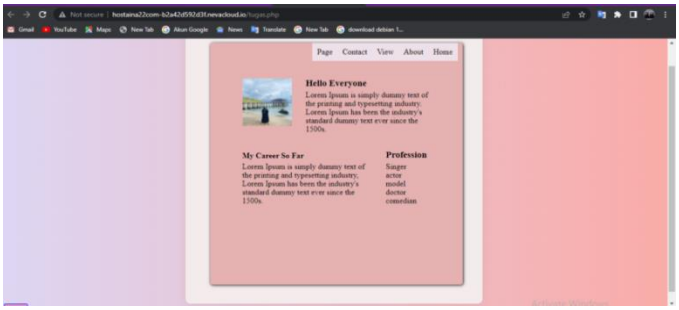
Gambar 4 menjelaskan bahwa ada tiga container image. Dimana container 1 terdapat *web server* 1 dengan port 81,

container 2 terdapat *web server* 2 dengan port 82 dan container 3 *web server* 3 dengan port 83.

III. HASIL DAN PEMBAHASAN

A. Kinerja Web Server

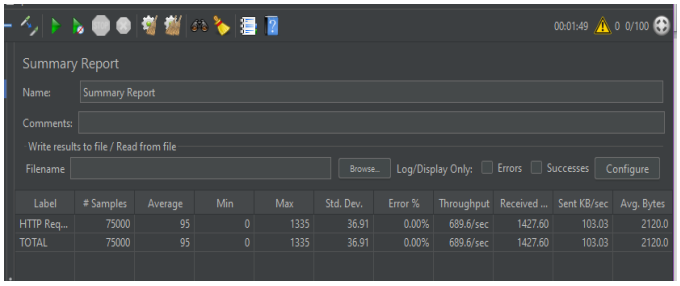
Ketika client mencoba akses ke web server melalui alamat domain load balancing, maka client akan menerima respon salah satu halaman web server.



Gambar 5 Tampilan Web Server

B. Pengujian

Apache jmeter merupakan aplikasi berbasis java bersifat *open source* dan digunakan untuk melakukan test seperti *load/stress testing web application*, dan FTP *Application* untuk mensimulasikan beban kerja pada jaringan, sekelompok *server*, *server* atau objek yang digunakan untuk menguji kekuatan kinerja secara keseluruhan dan menganalisanya jumlah *traffic* yang berbeda [4]. Adapaun tampilan jmeter sebagai berikut.



Gambar 6 Tampilan Jmeter

Berdasarkan gambar 6 *Summary Report* adalah salah satu jenis listener (pendengar) yang dapat digunakan dalam Apache JMeter untuk menganalisis hasil pengujian kinerja. Listener ini memberikan ringkasan data yang bermanfaat tentang kinerja pengujian. *Label* dari elemen pengujian yang sedang dieksekusi, seperti nama permintaan HTTP atau tindakan lain yang sedang diuji. *Samples* Jumlah total permintaan (sampel) yang telah dieksekusi selama pengujian. *Average* adalah rata-rata waktu yang dibutuhkan untuk menyelesaikan satu sampel (permintaan) dalam pengujian. Ini mengukur kinerja rata-rata.Min waktu tercepat yang dibutuhkan untuk menyelesaikan satu sampel dalam pengujian. *Max* waktu terlama yang dibutuhkan untuk menyelesaikan satu sampel dalam pengujian.Error persentase kesalahan yang terjadi selama pengujian. Ini mencakup permintaan yang menghasilkan respons kesalahan.

Pengujian performa mengunnakan *load balancing* dan tidak menggunakan *load balancing* dengan jumlah *traffic* 1,10,50,100,1000 . Berikut table perbandingan hasil pengujian pemberian beban pada *load balancing* dan tidak menggunakan *load balancing*.

TABEL 1
PERBANDINGAN LOAD BALANCING DAN TIDAK LOAD BALANCING DENGAN TRAFFIC 1

traffic	Looping	*Server*	Hasil	
			Throughput	error
1	1	*Load balancing*	11.1/sec	0.00%
		Tidak *load balancing*	8.6/sec	0.00%
	10	*Load balancing*	24.8/sec	0.00%
		Tidak *load balancing*	12.8/ sec	0.00%
	50	*Load balancing*	18.2 /sec	0.00%
		Tidak *load balancing*	16.4/ sec	0.00%
	100	*Load balancing*	148.8 /sec	0.00%
		Tidak *load balancing*	17.3/ sec	0.00%
	200	*Load balancing*	151.7/sec	0.00%
		Tidak *load balancing*	72.2/sec	0.00%
	500	*Load balancing*	156.1/sec	0.00%
		Tidak *load balancing*	151.0/ sec	0.00%
	750	*Load balancing*	176.7/sec	0.00%
		Tidak *load balancing*	150.1/ sec	0.00%
	1000	*Load balancing*	165.5/sec	0.00%
		Tidak *load balancing*	189.9/ sec	0.00%

Bedasarkan table 1 menjelaskan bahwa jumlah *traffic* 1 dengan looping 1, 10,50,100,200,500,750, dan 1000 maka nilai rata-rata *throughput load balancing* 14.57/sec dan tidak menggunakan *Load balancing* nilai rata-rata *throughput* 11.53/sec dan tidak ada mengalami error pada web *server*.

TABEL 2
PERBANDINGAN LOAD BALANCING DAN TIDAK LOAD BALANCING DENGAN TRAFFIC 10

traffic	Looping	*Server*	Hasil	
			Throughput	error
10	1	*Load balancing*	10.6 / sec	0.00%
		Tidak *load balancing*	10.2/sec	0.00%
	10	*Load balancing*	57.8 / sec	0.00%
		Tidak *load balancing*	56.7 /sec	0.00%
	50	*Load balancing*	133.7 /sec	0.00%
		Tidak *load balancing*	69.4/sec	0.00%
	100	*Load balancing*	148.8 /sec	0.00%
		Tidak *load balancing*	68.4/sec	0.00%
	200	*Load balancing*	151.7/sec	0.00%
		Tidak *load balancing*	72.2/sec	0.00%
	500	*Load balancing*	156.1/ sec	0.00%
		Tidak *load balancing*	151.0/sec	0.00%
	750	*Load balancing*	176.7/ sec	0.00%
		Tidak *load balancing*	150.1/sec	0.00%
	1000	*Load balancing*	189.9/ sec	0.00%
		Tidak *load balancing*	165.5/sec	0.00%

Bedasarkan table 2 menjelaskan bahwa jumlah *traffic* 10 dengan looping 1, 10,50,100,200,500,750, dan 1000 maka nilai rata-rata *throughput load balancing* 742.9/sec dan tidak menggunakan *Load balancing* rata-rata *throughput* 270,9 /sec dan tidak ada mengalami error pada web *server*.

TABEL 3
PERBANDINGAN LOAD BALANCING DAN TIDAK LOAD BALANCING DENGAN TRAFFIC 50

traffic	Looping	*Server*	Hasil	
			Throughput	error
50	1	*Load balancing*	34.1/sec	0.00%
		Tidak *load balancing*	18.2/sec	0.00%
	10	*Load balancing*	308.6 /sec	0.00%
		Tidak *load balancing*	132.0 /sec	0.00%
	50	*Load balancing*	506.2/sec	0.00%
		Tidak *load balancing*	171.4/ sec	0.00%
	100	*Load balancing*	299.4/sec	0.00%
		Tidak *load balancing*	192.4/ sec	0.00%
	200	*Load balancing*	523.6/sec	0.00%
		Tidak *load balancing*	277.4/sec	0.00%
	500	*Load balancing*	546.1/sec	0.00%
		Tidak *load balancing*	202.1/ sec	0.00%
	750	*Load balancing*	601.8sec	0.00%
		Tidak *load balancing*	293.0/ sec	0.00%
	1000	*Load balancing*	535.3/sec	0.00%
		Tidak *load balancing*	303.2/ sec	0.00%

Bedasarkan table 3 menjelaskan bahwa jumlah *traffic* 50 dengan looping 1, 10,50,100,200,500,750, dan 1000 maka nilai rata-rata *throughput load balancing* 403.9/sec dan tidak menggunakan *Load balancing* rata-rata *throughput* 189.83/sec dan tidak ada mengalami error pada web *server*.

TABEL 4
PERBANDINGAN LOAD BALANCING DAN TIDAK LOAD BALANCING DENGAN TRAFFIC 100

Traffic	Loop	*Server*	Hasil	
			Throughput	error
100	1	*Load balancing*	89.1/sec	0.00%
		Tidak *load balancing*	19.7/ sec	0.00%
	10	*Load balancing*	515.5/sec	0.00%
		Tidak *load balancing*	203.9/ sec	0.00%
	50	*Load balancing*	663.7/sec	0.00%
		Tidak *load balancing*	190.5/ sec	0.00%
	100	*Load balancing*	672.7/sec	0.00%
		Tidak *load balancing*	201.1/ sec	0.00%
	200	*Load balancing*	665.8/sec	0.00%
		Tidak *load balancing*	236.1/sec	0.00%
	500	*Load balancing*	594.8/sec	0.00%
		Tidak *load balancing*	312.1/ sec	0.00%
	750	*Load balancing*	689.6/sec	0.00%
		Tidak *load balancing*	442.4/ sec	0.00%
	1000	*Load balancing*	502.4/sec	0.00%
		Tidak *load balancing*	698.3/ sec	0.00%

Bedasarkan table 4 menjelaskan bahwa jumlah *traffic* 100 dengan looping 1, 10,50,100,200,500,750, dan 1000 maka nilai rata-rata *throughput load balancing* 569,73 /sec pada tidak menggunakan *Load balancing* nilai rata-rata 273.6/ sec dan tidak ada mengalami error.

TABEL 5
PERBANDINGAN LOAD BALANCING DAN TIDAK LOAD BALANCING DENGAN TRAFFIC 1000

traffic	Loopig	*Server*	Hasil	
	1		*throughput*	error
1000		*Load balancing*	205.4/sec	0.00%
	10	Tidak *load balancing*	79.7/ sec	0.00%
		Load balancing	817.9/sec	0.00%
	50	Tidak *load balancing*	139.4/sec	2.40%
		Load balancing	1046.8 /sec	0.00%
	100	Tidak *load balancing*	259.8/sec	0.00%
		Load balancing	1079.7/sec	0.03%

Bedasarkan table 5 menjelaskan bahwa jumlah *traffic* 100 dengan looping 1, 10,50,100, maka nilai rata-rata *throughput load balancing* 863.8/sec pada tidak menggunakan *Load balancing* nilai rata rata 991.2/sec dan ada mengalami error pada web *server*. Load balancing mengalami error 0.03 % dan tidak menggunakan load balancing mengalami error 0.635 % dengan *throughput* 354.3/ sec. Faktor terjadinya error karena jaringan kurang stabil disaat mengirim request, seharusnya jika jaringan stabil maka lebih akurat.

IV. KESIMPULAN

Adapun simpulan yang dapat penulis simpulkan setelah melakukan penelitian mengenai *Implementasi Sistem Load Balancing Web Server pada Jaringan public cloud computing menggunakan least* yaitu :

1. Bedasarkan analisis, dapat diperoleh hasil bahwa keberhasilan s*ystem* jalannya *traffic* secara real time yaitu 90 % dan hasil uji performa dari *web server* menggunakan aplikasi jmeter dengan jumlah *traffic* 1000 permintaan dalam satu waktu dengan looping 1,10,50 dan 100 pada load balancing nilai rata-rata *throughput* 630.2/sec dan tidak menggunakan load balancing nilai rata-rata *throughput* 354.5/sec.
2. Apabila salah satu jalur web server satu dimatikan , maka pengguna masih bisa tetap dapat mengakses web server dua dan web server tiga. Karena masih tersedia jalur yang aktif.
3. Penerapan sistem *load balancing* sudah sesuai dengan metode *least connection* dimana membagi beban bedasarkan banyaknya koneksi yang sedang dilayani oleh *server*.

REFERENSI

[1] Sakawiguna, dan ID CloudHost. 2019. *Pengertian Web server dan Fungsinya*. IDCloudHost.com diakses tanggal 7 maret 2023

[2] Apriliansyah, Fahmi, dkk. 2020. "Implementasi Load Balancing Pada Web server Menggunakan Nginx. Jurnal Teknologi dan Manajemen Informatika." Vol.6 No.1 Tahun 2020 P-I: 1693-6604 E-I: 2580-8044.

[3] Suhada. 2022. "Implementasi Load Balancing Menggunakan Metode Equal Cost Multi Path (Ecmp) Pada Kantor Bpn Lhokseumawe." Lhokseumawe: Politeknik Negeri Lhokseumawe.

[4] Khatami, Sukran. 2022. Implementasi Controller Access Point System Manager, Load Balancing Group Dan Access List Pada Jaringan Hotspot. Lhokseumawe: Politeknik Negeri Lhokseumawe.