Analisis Kinerja Algoritma *Weighted Round Robin* dan *Weighted Least Connection* pada Sistem *Load Balancing Web Server* dengan HAProxy Terintegrasi *Cacti Monitoring*

Teuku Muhammad Fathin Rifat[1], Unan Yusmaniar Oktiawati[1,*]
[1]Departemen Teknik Elektro dan Informatika, Sekolah Vokasi, Universitas Gadjah Mada;
tmfathin02@mail.ugm.ac.id
[*]Korespondensi: unan_yusmaniar@ugm.ac.id;

Abstract – The rapid advancement of technology over time has led to the high use of the internet in everyday life. The need for the internet affects website visitors who are increasing and makes the traffic load increase on the server. The more the amount of traffic to the server can cause the server to go down, if used with excessive amounts (overload). Therefore, it is necessary to improve the quality of the network that is directly related to the server as the role of a network traffic. So using a load balancing system is a solution to overcome the occurrence of server down. In the application process in this research, to overcome this problem, Nginx, HAproxy and a virtual machine system using virtualbox tools are needed for this. The Load Balancing method uses two servers and three servers, where this method will divert the traffic load from a full web server to another web server according to the number of web servers used. And will be monitored using Cacti by displaying graphical data from load balancing performance on the web server. The results show that the implementation of HAProxy on a web server load balancing system that uses the Weighted Round Robin and Weighted Least Connection algorithms can overcome problems caused by excessive traffic loads. The Weighted Least Connection algorithm provides superior values on the parameters of throughput, response time, request error, cpu usage, and memory usage compared to the Weighted Round Robin algorithm on the web server load balancing system.

Keywords – Web Server , Load Balancing, HAProxy, Cacti Monitoring

Intisari – Kemajuan teknologi yang semakin pesat dari waktu ke waktu menyebabkan tingginya penggunaan internet dalam kehidupan sehari-hari. Kebutuhan internet mempengaruhi pengunjung *website* yang semakin meningkat dan membuat beban *traffic* meningkat pada *server*. Semakin banyak jumlah *traffic* menuju *server* dapat menyebabkan *server* menjadi *down* jika digunakan dengan jumlah berlebih (*overload*). Oleh karena itu, dibutuhkan suatu peningkatan kualitas jaringan yang berhubungan langsung dengan *server* sebagai peran dari suatu lalu lintas jaringan. Penggunaan sistem *load balancing* merupakan solusi untuk mengatasi terjadinya *server down*. Penelitian ini memerlukan Nginx, HAproxy, dan sistem *virtual machine* menggunakan tools Virtualbox. Metode *load balancing* menggunakan dua *server* dan tiga *server*. Kedua metode ini akan mengalihkan beban *traffic* dari *web server* yang sudah penuh kepada *web server* lainnya sesuai dengan jumlah *web server* yang digunakan. Kemudian, selanjutnya akan dimonitoring menggunakan Cacti dengan menampilkan data grafik dari kinerja *load balancing* pada *web server*. Hasil menunjukkan dengan implementasi HAProxy pada sistem *load balancing web server* yang menggunakan algoritma *Weighted Round Robin* dan *Weighted Least Connection* dapat mengatasi permasalahan yang diakibatkan oleh beban *traffic* yang berlebihan. Algoritma *Weighted Least Connection* memberikan nilai yang lebih unggul pada parameter *throughput, response time, error request,* CPU *usage,* dan *memory usage* dibandingkan algoritma *Weighted Round Robin* pada sistem *load balancing web server*.

Kata kunci – *Web Server*, *Load Balancing*, HAProxy, *Cacti Monitoring*

I. PENDAHULUAN

Seiring berkembangnya teknologi komunikasi dan informasi, terutama dalam teknologi jaringan sangat berkaitan erat dengan teknologi internet yang menjadi salah satu sumber penyedia informasi terpopuler di era modern ini. Hampir seluruh sektor yang berada di sekitar telah memanfaatkan teknologi informasi yang menggunakan internet dalam membantu pekerjaan sehari-hari ataupun sekadar untuk hiburan semata. Perkembangan pengguna internet berpengaruh terhadap peningkatan data yang ditransmisikan. Paradigma di zaman sekarang telah berubah, salah satunya pada sektor hiburan dan informasi. Pada zaman dulu, sarana informasi berbentuk fisik memegang peranan penting dalam mendapatkan informasi terbaru. Saat ini hampir seluruh informasi yang dibutuhkan oleh khalayak umum baik informasi terdahulu hingga terkini, telah dibuat dan ditransmisikan ke dalam bentuk informasi digital yang dapat diakses kapan saja dan di mana saja dengan mudah. Tidak hanya pada sektor hiburan dan informasi, sektor perekonomian juga sangat memerlukan teknologi dalam mendongrak penjualan dan pendapatan. Terlebih dalam beberapa tahun belakangan, dunia dilanda wabah virus corona yang menyebabkan seluruh aktivitas di luar ruangan dibatasi. Salah satu yang memperoleh dampaknya adalah aktivitas berdagang. Hal tersebut menyebabkan para pedagang mengalami kerugian. Dengan demikian, untuk memenuhi target penjualan dan mendapatkan keuntungan, para pedagang beralih menggunakan teknologi untuk melanjutkan proses berdagang baik dengan membangun *website e-commerce* pribadi atau bergabung dengan *startup e-commerce*. Berdasarkan data pada Statista Market Insight, jumlah pengguna *e-commerce* baik secara *website* maupun aplikasi akan diproyeksikan mencapai 196,477 juta pengguna hingga akhir tahun 2023 [1].

Dalam mendukung pemenuhan kebutuhan untuk mendapatkan informasi terkini dan proses jual beli secara *online* diperlukan akses yang cepat dan stabil, salah satu aspek yang mempengaruhi kecepatan akses dan kestabilan sebuah alamat *website*, yaitu *server* penyedia layanan. Perangkat *server* atau disebut juga sebagai penyedia layanan merupakan sistem komputer yang memiliki layanan khusus berupa pengelolaan dan penyimpanan data. Perangkat *server* memiliki peran penting dalam menyediakan layanan akses yang lebih cepat untuk proses mengirim atau menerima data yang tersedia pada perangkat *server*. Apabila klien semakin banyak maka beban kerja kerja *server* semakin berat, sehingga diperlukan spesifikasi *server* yang bagus dan dapat melayani berbagai permintaan klien. Instansi perusahaan dan pengusaha individu yang menggunakan sistem informasi berbasis *website* dalam mengoperasikan segala kegiatan bisnisnya, membutuhkan *server* dengan spesifikasi yang cukup mumpuni, khususnya ketika banyak klien yang mengakses. Perangkat *server* dengan spesifikasi tinggi sangat diperlukan dalam membangun *webserver*, sehingga dapat memberikan layanan yang cepat dan stabil kepada klien ketika terdapat pengakses berjumlah banyak. Namun, biaya yang diperlukan dalam memenuhi kebutuhan tersebut tergolong tidak murah. Perangkat *server* tunggal yang memiliki spesifikasi cukup tinggi sangat rawan mengalami *trouble* dan *down* ketika mengalami lonjatan *request* sangat besar sehingga menyebabkan *website* tidak dapat diakses oleh pengunjung.

Penggabungan beberapa *server* menjadi satu kesatuan yang bekerja secara bersamaan (pemerataan beban *server*) merupakan salah satu solusi dalam mengatasi permasalahan diatas. Salah satu metode yang dapat digunakan yaitu *load balancing*. *Load balancing* merupakan suatu teknik dalam mendistribusikan beban trafik pada dua atau lebih jalur koneksi secara optimal, memaksimalkan *throughput*, memperkecil *response time*, dan menghindari terjadinya *overload* pada salah satu *server* [2], [3]. Tujuan dari *Load balancing web server,* yaitu meringankan beban yang ditanggung oleh setiap *server*, sehingga dapat meningkatkan kinerja *server* yang *high availability* dan terjaga meskipun salah satu *server* mengalami *down* dan tidak dapat melayani *request* dari klien, dikarenakan *server* lain secara otomatis akan menggantikannya. Salah satu aplikasi yang dapat digunakan untuk sistem *load balancing web server* adalah HAProxy. HAProxy atau *High Availibility Proxy* merupakan perangkat lunak *open-source* yang dikembangkan khusus untuk melakukan *load balancing* dan biasanya digunakan dalam penanganan lalu lintas HTTP dan TCP yang sangat cepat, seperti aplikasi web dengan tingkat lalu lintas tinggi dan telah digunakan oleh *website* terkenal, seperti Github, Stackoverfow, Reddit, Tumblr, Instagram, dan Twitter [4], [5]. Terdapat beberapa algoritma *load balancing* pada HAProxy, antara lain *Weighted Round Robin* dan *Weighted Least Connection*. Algoritma *Weighted Round Robin* merupakan perbaikan dari algoritma *Round Robin* yang mendistribusikan beban secara seimbang menuju seluruh *server* tanpa memperhatikan kapasitas proses yang berbeda. Masing-masing *server* diberi bobot dengan bilangan *interger* yang menunjukkan kapasitas proses, di mana bobot awal adalah 1. Algoritma *Weighted Least Connection* mengarahkan koneksi dari jaringan ke *server* yang mempunyai jumlah koneksi aktif paling sedikit. Algoritma ini merupakan algoritma dinamis karena menghitung jumlah koneksi yang aktif pada setiap *server* secara dinamis dan algoritma ini cukup baik untuk memperhalus distribusi koneksi saat *request* sangat beragam [6], [7].

Berdasarkan hal di atas, penelitian ini melakukan "Analisis Kinerja Algoritma *Weighted Round Robin* dan *Weighted Least Connection* pada Sistem *Load Balancing Web Server* dengan HAProxy Terintegrasi *Cacti Monitoring*". Penelitian ini menggunakan dua buah algoritma, yaitu *Weighted Round Robin* dan *Weighted Least Connection*. Tujuan penelitian ini untuk mengetahui kinerja dari sistem *load balancing* yang menggunakan dua buah algoritma tersebut, kemudian menemukan algoritma mana yang memberikan hasil terbaik dari segi nilai *throughput*, *response time*, dan *error request* serta memonitoring dari kinerja *load balancing* dengan *Cacti Monitoring*.

II. DASAR TEORI

Suatu penelitian diperlukan adanya hasil dari beberapa penelitian sebelumnya sebagai landasan teori dan acuan dalam penelitian ini. Penelitian [8] merancang sistem *load balancing* pada *web server* menggunakan metode *apache* yang diimplementasikan sebagai *load balancing* serta sebagai *web server* yang memiliki tujuan dalam meningkatkan kinerja *web server*. Pada sistem *load balancing*-nya, mengimplementasikan teknik *reverse proxy* dengan konfigurasi yang diberikan berupa *method by busyness* dan *method by request*. Skenario pengujian dibagi menjadi tiga skenario pengujian, yaitu tes terhadap koneksi antara *database server* dengan *web server,* tes sistem kerja *load balancer method by request,* tes sistem kerja *load balancer method by busyness*. Penelitian ini menyatakan bahwa *sistem load balancing* metode *apache* dengan *method by request* akan melanjutkan seluruh *request* yang masuk menuju ke beberapa *web server* secara berurutan, dan *load balancing* metode *apache* dengan *method by busyness* akan mengarahkan menuju *web server* yang memiliki waktu respon tinggi.

Selain dengan Apache*, load balancing* dapat diterapkan dengan menggunakan Nginx yang lebih ringan dan menjadi *server* khusus *reverse proxy* tanpa membebani *hardware* yang digunakan sebagai *load balancer* [8], [9]. Penelitian [10] menggunakan layanan Nginx sebagai *load balancing web server* yang dibangun pada *virtual machine* VirtualBox. Pengujian dilakukan dengan menggunakan serangan kiriman paket secara simultan pada waktu beberapa detik dengan *tools siege.* Hasil menunjukkan bahwa dengan menggunakan Nginx untuk *load balancing web server* dapat membantu *web server* mengalihkan *request* dari *client* ke beberapa *web*

server secara bergantian ketika terjadi *overload*, tetapi secara manual.

Penelitian [11] merancang implementasi *load balancing* dengan tambahan metode NTH. Penelitian ini memiliki infrastruktur menggunakan satu komputer *server*, Mikrotik RB941-2nD-TC, dua komputer *client*, satu *switch,* dua NIC dengan dukungan *software ubuntu server, web server, Openssh Server, Winbox,* dan *WinSCP.* Peneliti menggunakan metode *load balancing* NTH yang didukung dengan NIC sebagai jalur dalam pembagi beban yang ditanggung. Hasil dari penelitian berikut menunjukkan bahwa dengan menggunakan metode *load balancing NTH* berhasil yang ditandai dengan keseimbangan pada proses *download file* dan pada kecepatan *download 2 client* sama besar yakni 256 kBps sesuai dengan konfigurasi *bandwidth* yang telah dibuat yaitu sebesar 512 kBps.

Penelitian selanjutnya menggunakan metode *clustering* server untuk proses *load balancing*. Menerapkan *clustering server* pada sistem *load balancing* dapat meningkatkan kinerja seluruh *server* dan mendukung sinkronisasi file-file yang disimpan pada seluruh *server* [12]. *Server* yang digunakan sebagai *web server* dibuat menjadi sebuah *cluster* dengan menggunakan fungsi *failover cluster* dan *load balancing cluster* yang memungkinkan *server-server* tersebut berkolaborasi dan saling bekerja sama dalam menciptakan kinerja yang maksimal, sehingga jika salah satu *server* mengalami masalah, *server* lain dapat mengambil alih tugasnya dan menjaga kualitas layanan. Hasil dari penelitian tersebut menunjukkan bahwa dengan menggunakan *clustering server* performa yang dihasilkan untuk layanan sangat baik dan berjalan sesuai harapan, terutama pada sinkronisasi file. Dengan konfigurasi yang dirancang sedemikian rupa dan juga memanfaatkan *clustering server*, file-file dapat tersimpan otomatis pada ketiga *server* meskipun proses penyimpanan file hanya dilakukan pada salah satu *server*.

Penelitian [13] merancang sistem *load balancing* terhadap *web server* dengan menggunakan Nginx. Sistem yang dirancang menggunakan sebagai *web server* dan *load balancer* serta menerapkan algoritma *round robin*, algoritma *least connection*, dan algoritma *IP hash* pada sistem *load balancing.* Penelitian ini melakukan pengujian fungsionalitas untuk mengetahui kemampuan sistem yang dirancang dalam mengatasi masalah yang ada dan untuk mengetahui algoritma mana yang dapat memberikan hasil terbaik pada sistem *load balancing* di antara ketiga algoritma yang digunakan.

Penelitian [14] memiliki tujuan untuk menguji bagaimana *load balancing* dengan HAProxy yang menggunakan algoritma *round robin* dapat memberikan kinerja terbaik dalam mengatasi masalah ketika salah satu *server down* atau mengalami *trouble*. Penelitian ini menyatakan bahwa penerapan sistem *load balancing* menggunakan HAProxy dapat meningkatkan ketersediaan *server* (*uptime*) pada suatu *website* dan memberikan kualitas yang terjaga dan stabil ketika salah satu *server* mengalami *down*, sehingga *user* tidak merasakan terjadi *down time*.

Penelitian [15] mengimplementasikan dua buah metode *load balancing* pada pengujian terhadap sistem *e-learning* yang dibandingkan nilainya dari beberapa aspek pengujian, seperti nilai *throughput* dan *response time*. Hasil dari penelitian ini menunjukkan bahwa *load balancing* yang mengimplementasikan metode HAProxy memberikan kinerja yang sangat baik dibandingkan dengan *load balancing* yang mengimplementasikan metode Nginx dengan hasil pada nilai *response time* dan *throughput* HAProxy meraih hasil yang sangat baik sebesar 533,14 ms untuk *response time* dan 40,84 kB/sec untuk *throughput*.

Penelitian yang [16] memiliki tujuan untuk menguji bagaimana algoritma penjadwalan *Weighted Least Connection* yang telah diimprovisasi dapat menjawab kebutuhan *load balancing* pada *web cluster system* dengan lebih baik lagi dibanding dengan algoritma *Weighted Least Connection* sebelum diimprovisasi. Penelitian ini menggunakan dua buah protokol, yaitu *server selection* dan *overload* dengan tujuan membantu proses *load balancing* terhadap *web cluster system* yang menggunakan *Weighted Least Connection* sebagai algoritma pada sistem *load balancing*. Hasil penelitian ini menunjukkan bahwa hasil improvisasi dari penggunaan *Weighted Least Connection* yang menerapkan dua buah protocol baru mampu menghasilkan kinerja yang lebih baik pada sistem *load balancing* terhadap *web cluster.*

Penelitian [17] memiliki tujuan untuk menguji bagaimana kinerja algoritma *least connection* dan IP *hash* pada sistem *load balancing web server* yang melalui jaringan SDN. Kedua algoritma yang digunakan pada penelitian ini akan dievaluasi dengan beberapa parameter dasar untuk *web server*. Hasil penelitian ini menunjukkan bahwa algoritma IP *hash* menghasilkan kinerja yang baik, dengan hasil pada parameter *response time* sebesar 61 ms, hasil pada *throughput* sebesar 1,23 Mbps, dan hasil pada *resource memory* sebesar 189,2 MB.

Penelitian sebelumnya masih menggunakan Nginx, HAProxy, dan Apache sebagai metode *load balancing*. Selain dengan metode tersebut, *load balancing* juga dapat menggunakan metode Traefik and Envoy. Penelitian [18] menggunakan tiga buah metode, yaitu HAProxy, Nginx, dan Traefik and Envoy untuk proses *load balancing* pada *web server*. Hasil dari penelitian ini menunjukkan bahwa metode Traefik and Envoy memiliki kinerja yang cukup bagus, lebih bagus dari pada Nginx, tetapi tidak sebagus HAProxy, dengan data menunjukkan bahwa *traffic* berkinerja 24,1% lebih baik, *envoy* 26,9% lebih baik, dan HAProxy 36,0% lebih baik dibandingkan Nginx.

III. METODOLOGI

A. Alat

Alat yang digunakan pada penelitian ini berupa laptop dan *Virtual Machine* memiliki spesifikasi seperti pada Tabel 1 dan Tabel 2.

- Laptop

Tabel 1. Spesifikasi Laptop

Spesifikasi	Detail
Sistem Operasi	Windows 11 Home Single Language 64-bit
Processor	11th Gen i7-11800H 2.30GHz
Memory	16GB RAM
GPU	NVIDIA GeForce RTX 3050Ti
Penyimpanan Sistem	512GB SSD NVME M.2

- *Virtual Machine*

Tabel 2. Spesifikasi *Virtual Machine*

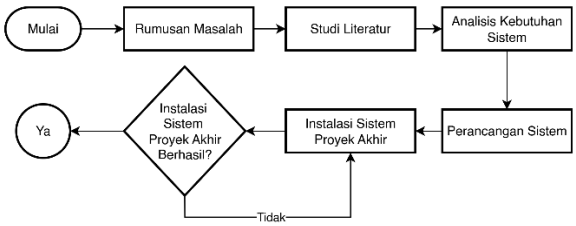
Virtual Machine	Spesifikasi	
Load Balancer	OS	Debian 11.6
	RAM	4 GB
	Disk	80 GB
	Alamat IP	
	Nama VM	Load Balancer
Web Server 1	OS	Debian 11.6
	RAM	2 GB
	Disk	50 GB
	Alamat IP	
	Nama VM	Nginx Web Server
Web Server 2	OS	Debian 11.6
	RAM	2 GB
	Disk	50 GB
	Alamat IP	
	Nama VM	Nginx Web Server
Web Server 3	OS	Debian 11.6
	RAM	2 GB
	Disk	50 GB
	Alamat IP	
	Nama VM	Nginx Web Server

B. Bahan

Dalam penelitian ini menggunakan beberapa bahan seperti *Virtual Machine* (VirtualBox), Windows OS, Web Browser.

C. Diagram Alir Penelitian

Secara ringkas alur penelitian secara singkat dapat dilihat pada Gambar 1.



Gambar 1. Diagram Alir Penelitian

D. Pengujian

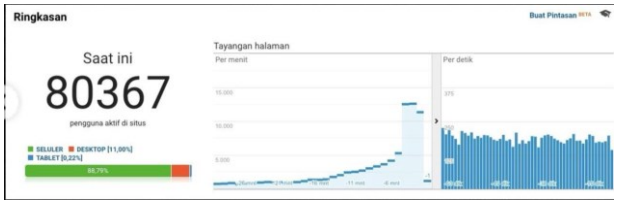
Pengujian pada penelitian berikut ini dilakukan dengan menguji sistem kinerja dari *load balancing* HAProxy terhadap *webserver* yang menggunakan dua buah algoritma, yaitu *Weighted Round Robin* dan *Weighted Least Connection*. Penggunaan dua buah algoritma memiliki tujuan untuk melihat algoritma mana yang dapat memberikan kualitas terbaik pada sistem *load balancing* terhadap *webserver*. Dalam pengujian berikut, menggunakan dua buah variasi jumlah *server* dengan pengujian pertama menggunakan tiga buah *server* dan pengujian kedua menggunakan dua buah *server.* Hal tersebut bertujuan agar dapat mengidentifikasi terhadap kemampuan sistem *load balancing* dalam menangani lalu lintas dengan baik ketika terdapat lebih banyak atau lebih sedikit sumber daya *server* yang tersedia serta memastikan bahwa sistem *load balancing* dan infrastruktur web dapat beroperasi secara optimal dalam berbagai kondisi yang mungkin terjadi di lingkungan produksi.

Pengujian berikut akan mengirimkan *request* atau permintaan dengan dua buah variasi yang terdiri dari dua buah jumlah *request*, pertama dengan jumlah terendah sebesar 7.500 *request* dan kedua dengan jumlah terbesar sebesar 10.000 *request*. Jumlah *request* ditetapkan sebanyak dua buah dengan tujuan untuk menguji performa dari infrastruktur dalam menghadapi lonjakan lalu lintas atau beban yang tidak terduga baik dengan jumlah *request* yang sangat sedikit maupun dengan jumlah *request* yang sangat banyak.

Selain itu, hal tersebut juga bertujuan untuk lebih memahami secara lebih dari kemampuan sistem, mengidentifikasi potensi *bottleneck* atau masalah kinerja, serta memastikan bahwa infrastruktur web dapat mengatasi beban lalu lintas yang bervariasi dengan baik dalam lingkungan produksi yang sebenarnya. Waktu yang digunakan pada proses pengiriman seluruh jumlah, jumlah *request* sebesar 7.500 dan jumlah *request* sebesar 10.000 menggunakan waktu selama 10s agar pemantauan yang dilakukan sesuai dengan keadaan yang terjadi.

Hal ini dikarenakan *use case* yang digunakan adalah *website* brand lokal ketika *launching* produk *limited edition* terbaru di mana *client* yang mengakses ketika ingin membeli produk tersebut berkisar 300 *client* per detik. Pengambilan jumlah *request* yang diujikan berdasarkan contoh pada suatu *website brand* pakaian lokal ketika merilis suatu produk

terbaru yang hanya memiliki stok terbatas, seperti yang tertera pada Gambar 2.



Gambar 2. Jumlah *Client Website Brand Heymale*

IV. HASIL DAN PEMBAHASAN

Penelitian ini menghasilkan analisis kinerja dari dua buah algoritma *load balancing* yang digunakan. Dengan membandingkan dua buah algoritma memberikan gambaran kepada *user* yang menggunakan *load balancing* baik untuk kegunaan pribadi maupun untuk perusahaan. Agar sistem yang digunakan dapat berjalan dengan lancar dan mengatasi masalah, terutama mengenai masalah *server* yang *down* ketika *overload traffic*, pengujian performa dilakukan dengan beberapa konfigurasi baik pada sisi jumlah *server* yang hidup maupun pada sisi algoritma *load balancing* dengan *request* atau permintaan yang akan dikirimkan pada pengujian berjumlah 7.500 *request* dan 10.000 request dan berlaku pada setiap pengujian.

A. Analisis Hasil Pengujian WRR dan WLC dengan Tiga Buah *Webserver*

Pada pengujian ini dilakukan dengan tiga buah konfigurasi *weight* pada setiap *webserver* yang mana setiap *webserver* memiliki jumlah *weight* yang berbeda. Konfigurasi *weight* pertama memiliki perbandingan 3 : 1 : 2, dimana *webserver* pertama memiliki bobot *weight* yang lebih besar agar dapat menerima lebih banyak beban, sedangkan pada *webserver* kedua memiliki bobot *weight* yang lebih kecil agar beban yang diterima tidak telalu besar, dan pada *webserver* ketiga memiliki bobot *weight* yang tidak terlalu besar dan tidak terlalu kecil atau berada diantara beban *webserver* pertama dan *webserver* kedua sehingga kinerja beban yang diterima tidak terlalu berat dan tidak terlalu ringan. Pada konfigurasi *weight* kedua memiliki perbandingan 2 : 5 : 3, dimana *webserver* kedua memiliki bobot yang lebih besar, *webserver* pertama memiliki bobot yang lebih kecil, dan *webserver* ketiga memiliki jumlah bobot di antara *webserver* pertama dan *webserver* kedua. Pada konfigurasi *weight* ketiga memiliki perbandingan 2 : 1 : 3, dimana *webserver* ketiga memiliki bobot yang lebih besar dibandingkan dengan kedua *webserver* lainnya. Jumlah *request* yang akan dikirimkan menuju *load balancer* sesuai yang telah ditetapkan dengan dua buah *request*, yaitu 7.500 *request* dan 10.000 *request*. Hasil yang didapatkan ditampilkan dalam bentuk tabel.

1. Analisis Hasil Nilai Pengujian *Throughput*

Pengujian nilai *throughput* pada penelitian berikut dilakukan dengan menjalankan beberapa penelitian yang terdiri dari penelitian menggunakan *Weighted Round Robin* dan *Weighted Least Connection*. Hasil Data didapatkan dari proses pengujian dengan tools JMeter. Untuk nilai hasil pengujian *throughput* pada penelitian berikut dapat terlihat pada Tabel 3.

Tabel 3. *Throughput* Tiga *Web Server*

Banyak *request*	*Throughput*					
	Weighted Round Robin			*Weighted Least Connection*		
	3:1:2	**2:5:3**	**2:1:3**	**3:1:2**	**2:5:3**	**2:1:3**
7.500	1059,8	997,9	998,5	999,5	1.000	999,1
10.000	997,7	870,1	999,1	1018,8	995,6	1.000

Berdasarkan tabel hasil pengujian nilai *throughput* menunjukkan bahwa nilai *throughput* yang didapatkan dari penggunaan *weighted least connection* memiliki kinerja yang lebih baik dibandingkan *throughput* yang dihasilkan oleh *weighted round robin* pada tiga buah pengujian dengan nilai yang didapatkan oleh WLC sebesar 1018,8 Kbits/s, 1.000 Kbits/sec, dan 995,6 Kbits/s. Namun, pada salah satu pengujian algoritma WRR mendapatkan hasil lebih baik dibanding WLC dengan nilai sebesar 1059,8 Kbits/sec. Berdasarkan data pengujian *throughput* yang dihasilkan, algoritma *weighted least connection* mampu mengalokasikan jumlah permintaan atau jumlah koneksi secara baik dan mampu menghindari terjadinya *overload* permintaan atau *request* pada satu atau beberapa *webserver* yang ada.

2. Analisis Hasil Nilai Pengujian *Response Time*

Pengujian *Response Time* dilakukan dengan tujuan mengetahui besaran kecepatan respons *server* dalam menanggapi *request* klien. Pengujian dilakukan menggunakan JMeter dengan jumlah *request* sebesar 7500 dan 10000. Untuk nilai hasil pengujian *response time* pada penelitian ini dapat terlihat pada Tabel 4.

Tabel 4. *Response Time 3 Web Server*

Banyak *request*	*Throughput*					
	Weighted Round Robin			*Weighted Least Connection*		
	3:1:2	**2:5:3**	**2:1:3**	**3:1:2**	**2:5:3**	**2:1:3**
7.500	63	5	12	8	5	9
10.000	29	12	61	14	9	8

Berdasarkan data pada hasil pengujian *response time* yang tertera pada tabel menunjukkan bahwa *response time* yang didapatkan oleh *weighted least connection* memiliki kinerja yang lebih baik dan cepat dibandingkan dengan *Weighted Round Robin*. Dengan rata-rata yang dihasilkan ketika menggunakan *weighted least connection* sebesar 9ms, algoritma *Weighted Least Connection* memiliki *response time* lebih baik, dikarenakan *request* dapat diteruskan ke *server* dengan optimal sesuai bobot masing-masing *server* dan dialihkan menuju *server* yang memiliki koneksi sedikit sehingga beban kerja *server* menjadi lebih merata. *Response*

time yang baik juga didukung oleh jumlah *request error* yang terjadi sangat sedikit.

3. Analisis Hasil Nilai Pengujian *Error Request*

Pengujian nilai *Error Request* dilakukan dengan tujuan untuk melihat algoritma yang mana memiliki performa lebih baik dalam mendistribusikan *request* menuju *server-server* yang ada. Data pengujian diambil dengan beberapa pengujian yang menggunakan tools JMeter. Untuk nilai dari hasil pengujian *error request* dapat terlihat pada Tabel 5.

Tabel 5. *Error Request 3 Web Server*

Banyak *request*	*Error Request (%)*					
	Weighted Round Robin			*Weighted Least Connection*		
	3:1:2	2:5:3	2:1:3	3:1:2	2:5:3	2:1:3
7.500	0	0	0	0	0	0
10.000	27,23	5,55	28,19	0,05	12,47	0

Berdasarkan tabel hasil pengujian *error request* menunjukkan bahwa algoritma *weighted least connection* menghasilkan performa yang sangat baik dalam menimalisir *error* yang terjadi. Adanya *Error* pada penggunaan *weighted least connection* lebih sedikit dibanding *error* yang terjadi pada penggunaan *weighted round robin,* seperti yang tertera pada tabel hasil. Rata-rata yang dihasilkan oleh *weighted least connection* pada pengujian dengan konfigurasi *weight* pertama lebih sedikit dibanding *weighted round robin* dan selisih rata-rata antara keduanya sebesar 25,98%. Dengan hasil tersebut menujukkan bahwa *weighted least connection* lebih baik dalam hal mendistribusikan atau membagi *request* yang datang sehingga *error* yang terjadi dapat diminimalisir.

B. Analisis Hasil Pengujian *Weighted Round Robin* dengan Dua Buah *Webserver*

Berbeda dengan pengujian pertama, pengujian kedua dilakukan dengan mematikan salah satu *webserver* agar dapat melihat dan memahami lebih lagi terkait kinerja apabila hanya memiliki dua buah *webserver* atau pada suatu kondisi salah satu *webserver* tiba-tiba *down* dan hanya memiliki beberapa *webserver* yang tersedia. Konfigurasi yang diberikan pada dua buah *webserver* juga tidak jauh dari sebelumnya, hanya berbeda pada jumlah beban atau *weight* pada *webserver* yang digunakan. Konfigurasi *weight* pertama memiliki perbandingan 3 : 1, di mana *webserver* pertama memiliki bobot *weight* yang lebih besar agar dapat menerima lebih banyak beban, sedangkan pada *webserver* kedua memiliki bobot *weight* yang lebih kecil agar beban yang diterima tidak terlalu besar atau lebih kecil dibanding *webserver* pertama. Pada konfigurasi *weight* kedua memiliki perbandingan 1 : 2, dimana *webserver* kedua memiliki bobot yang lebih besar, dan *webserver* pertama memiliki bobot yang lebih kecil.

1. Analisis Hasil Nilai Pengujian *Throughput*

Data hasil pengujian nilai *throughput* yang diambil pada penelitian ini dilakukan dengan menjalankan beberapa penelitian yang terdiri dari penelitian menggunakan *weighted round robin* dan *weighted least connection*. Data yang didapatkan dari proses pengujian dengan tools JMeter. Terkait nilai dari hasil pengujian *throughput* pada penelitian ini dapat terlihat pada Tabel 6.

Tabel 6. *Throughput 2 Web Server*

Banyak *Request*	*Throughput*			
	Weighted Round Robin		*Weighted Least Connection*	
	3:1	1:2	3:1	1:2
7.500	987,7	999,6	999,9	999,7
10.000	870,1	997,4	1.000	998,3

Berdasarkan data dari hasil pengujian penggunaan *throughput* yang tertera pada tabel menunjukkan bahwa *throughput* yang dihasilkan dari penggunaan *weighted least connection* pada pengujian dengan dua buah *webserver* memiliki performa yang baik dibandingkan *throughput* yang dihasilkan oleh algoritma *Weighted Round Robin* meskipun dengan selisih diantara setiap pengujian sebesar 0,01%. Berdasarkan data pengujian *throughput* yang dihasilkan bahwa algoritma *weighted least connection* dan *Weighted Round Robin* mampu mengalokasikan jumlah permintaan atau jumlah koneksi secara baik dan mampu menghindari terjadinya *overload* permintaan atau *request* pada satu atau beberapa *webserver* yang ada. Meskipun *weighted least connection* memiliki keunggulan sebesar 0,01% pada setiap pengujian.

2. Analisis Hasil Nilai Pengujian *Response Time*

Pengujian *Response Time* dilakukan dengan tujuan mengetahui besaran kecepatan *response server* dalam menanggapi *request* klien. Pengujian dilakukan menggunakan JMeter dengan jumlah *request* sebesar 7.500 dan 10.000. Hasil dari pengujian nilai *response time* pada penelitian ini dapat terlihat pada Tabel 7.

Tabel 7. *Response Time 2 Web Server*

Banyak *Request*	*Response Time (ms)*			
	Weighted Round Robin		*Weighted Least Connection*	
	3:1	1:2	3:1	1:2
7.500	7	6	6	6
10.000	9	9	9	8

Berdasarkan data dari hasil pengujian nilai *response time* yang tertera pada tabel menunjukkan bahwa nilai *response time* yang didapatkan oleh sistem *load balancing* dengan menerapkan *weighted least connection* dan *weighted round*

robin memiliki kinerja yang cukup baik pada pengujian terhadap dua buah *webserver* dengan hasil kedua nya hampir sama baik. Akan tetapi, secara keseluruhan *weighted least connection* memiliki hasil yang lebih unggul dibanding *weighted round robin*. *Weighted Least Connection* memiliki nilai *response time* lebih baik dan lebih cepat sebesar 1ms dibanding *weighted round robin* pada dua buah pengujian yang dilakukan.

3. Analisis Hasil Nilai Pengujian *Error Request*

Pengujian nilai *Error Request* dilakukan untuk melihat algoritma yang mana memiliki performa lebih baik dalam mendistribusikan *request* menuju *server-server* yang ada. Data pengujian diambil dengan beberapa pengujian yang menggunakan tools JMeter. Untuk nilai dari hasil pengujian nilai *error request* pada penelitian berikut dapat dilihat pada Tabel 8.

Tabel 8. *Error Request 2 Web Server*

Banyak *Request*	***Error Request (%)***			
	Weighted Round Robin		***Weighted Least Connection***	
	3:1	**1:2**	**3:1**	**1:2**
7.500	0	0	0	0
10.000	0	0,05	0	0,01

Berdasarkan tabel hasil pengujian *error request* menunjukkan bahwa algoritma *weighted least connection* pada pengujian dengan dua *webserver* menghasilkan performa yang sangat baik dalam menimalisir *error* yang terjadi dengan rata-rata error sebesar 0,01%. *Error* yang terjadi pada penggunaan *weighted least connection* lebih sedikit dibanding *error* yang terjadi pada pengunaan *weighted round robin,* seperti yang tertera pada tabel hasil. Selisih dari rata-rata yang dihasilkan oleh kedua algoritma tidak terlalu jauh dengan algoritma *weighted least connection* lebih unggul. Dengan hasil tersebut menunjukkan bahwa *weighted least connection* lebih baik dalam hal mendistribusikan atau membagi *request* yang datang sehingga *error* yang terjadi dapat diminimalisir.

C. Analisis Hasil Pengujian dengan Skema Lonjakan beban dan Pola Acak

Pengujian berikut menggunakan skema lonjakan beban dan pola acak dilakukan untuk mengetahui besaran nilai *response time* dan *error request* pada system load balancing dengan menggunakan algoritma *weighted round robin* dan *weighted least connection* dalam menghadapi kondisi yang sering terjadi di dunia nyata. Hasil dari pengujian nilai *response time* dan *error request* pada penelitian ini dapat terlihat pada Tabel 9 dan Tabel 10.

Tabel 9. *Response Time* dan *Error Request* Lonjakan Beban

Skema Lonjakan Beban		
Algoritma	***Response Time***	***Error Request***
Weighted Round Robin	490 ms	5,2%
Weighted Least Connection	260 ms	1,1%

Tabel 10. *Response Time* dan *Error Request* Pola Acak

Skema Pola Acak		
Algoritma	***Response Time***	***Error Request***
Weighted Round Robin	310 ms	3,5%
Weighted Least Connection	190 ms	0,7%

Berdasarkan hasil yang tertera pada Tabel 9 dan Tabel 10 menunjukkan bahwa algoritma *weighted least connection* mampu menghasilkan performa lebih baik pada pengujian dengan skema lonjakan beban maupun pola acak. Dengan selisih terhadap algoritma *weighted round robin* sebesar 150 ms hingga 250 ms untuk nilai *response time*-nya, sedangkan untuk selisih pada parameter *error request* sebesar 3% hingga 4%. Algoritma *weighted least connection* mendapatkan hasil performa seperti pada tabel di atas, dikarenakan sistem pembagian bebannya mempertimbangkan jumlah koneksi aktif pada setiap *server*. *Server* yang memiliki jumlah koneksi aktif sedikit atau kecil akan dikirimkan *load* beban yang banyak, sedangkan *server* yang memiliki jumlah koneksi aktif besar akan dikirimkan *load* beban yang sedikit, dan hal tersebut membuat distribusi pembagian beban lebih seimbang.

D. Analisis Hasil Pengujian *CPU Usage* dan *Memory Usage*

Data hasil pengujian *Memory Usage* diambil dengan melakukan dua buah pengujian dengan mengaktifkan *proxy* dan menonaktifkan *proxy* terhadap *single server* dan *multi server.* Untuk hasil pengujian *Memory Usage* pada penelitian ini dapat dilihat pada Tabel 11.

Tabel 11. CPU *Usage* dan *Memory Result*

Status Proxy	***Memory (MB)***	
	Single Server	***Multi Server***
Proxy Aktif	160 MB	210 MB
Proxy Non-Aktif	100 MB	130 MB
Persentase	60%	61,50%

Berdasarkan hasil yang tertera pada Tabel 11, hasil dari pengujian *memory usage* pada sistem yang menonaktifkan *proxy* menghasilkan performa lebih baik dalam penggunaan *memory* dibandingkan sistem yang mengaktifkan *proxy*. Pada sistem yang mengaktifkan *proxy* terhadap *single server* menghasilkan penggunaan *memory* sebesar 160 MB lebih

besar 60% daripada sistem yang menonatifkan *proxy* dengan hasil yang didapatkan sebesar 100 MB. Pada pengujian kedua dengan *multi server*, sistem yang mengaktifkan *proxy* menggunakan *memory* sebsar 210 MB dengan presentasi 61,50% lebih besar dari pada penggunaan *memory* pada sistem yang tidak mengaktifkan *proxy*. Peningkatan penggunaan *memory* pada sistem yang mengaktifkan *proxy* disebabkan oleh *overhead* tambahan dari manajemen sesi, inspeksi *header*, dan fitur-fitur lainnya yang berjalan pada *layer* 7.

Meskipun hasil pengujian menunjukkan bahwa algoritma *weighted least connection* memiliki kinerja yang lebih unggul dalam berbagai skenario pengujian, tetapi penting untuk memahami bahwa pemilihan algoritma *load balancing* juga bergantung pada konteks operasional dan tujuan sistem. *Weighted Least Connection* dalam hal adaptivitas terhadap beban koneksi aktif. Algoritma ini memonitor jumlah koneksi secara *real-time*, *weighted least connection* dapat mendistribusikan beban secara lebih seimbang ke *server* yang memiliki kapasitas tersedia lebih besar. Namun, keunggulan ini disertai dengan kompleksitas tambahan dalam proses monitoring koneksi aktif, yang dapat meningkatkan beban kerja pada sistem *load balancer*, terutama pada skala infrastruktur yang besar.

Sementara itu, algoritma *Weighted Round Robin* menawarkan kemudahan dalam konfigurasi dan efisiensi sistem, karena pembagian beban dilakukan berdasarkan bobot statis yang telah ditentukan sebelumnya tanpa memperhitungkan kondisi koneksi aktif saat itu. Algoritma ini sangat cocok digunakan pada sistem dengan spesifikasi *server* yang homogen serta beban *traffic* yang relatif stabil.

Namun, algoritma ini tidak cocok digunakan jika beban *traffic*-nya dinamis, karena dapat menyebabkan ketidakseimbangan distribusi beban yang dapat berdampak pada meningkatnya waktu respons dan tingkat eror. Dengan demikian, *trade-off* utama antara kedua algoritma terletak pada aspek kesederhanaan versus adaptivitas. Pemilihan algoritma *load balancing* yang tepat sebaiknya disesuaikan dengan karakteristik sistem dan kebutuhan operasional.

Meskipun algoritma *Weighted Least Connection* menunjukkan kinerja unggul dalam lingkungan dinamis, beberapa studi seperti [17] menunjukkan bahwa algoritma *Weighted Round Robin* tetap relevan dalam lingkungan server homogen dengan beban stabil, karena proses distribusinya lebih ringan dan deterministik. Selain itu, studi terbaru [18] dan [19] menyajikan perkembangan algoritma dan aplikasi *load balancing* modern, termasuk evaluasi performa HAProxy dibanding platform lain seperti Nginx, Traefik, dan Envoy, serta penerapan algoritma adaptif berbasis optimasi dan *machine learning*.

V. SIMPULAN

Melalui penelitian ini, ditarik kesimpulan yaitu *Load Balancing webserver* dengan HAProxy yang menggunakan dua buah algoritma berhasil dilakukan. Kemudian, pada pengujian *throughput* algoritma *weighted least connection* hampir secara keseluruhan memiliki hasil yang lebih bagus dengan hasil terbaik sebesar 1018,6 Kbits/s dan pada algoritma *Weighted Round Robin throughput* terbaik sebesar 1059,8. Pada parameter *response time*, algoritma *weighted least connection* memberikan hasil yang cukup maksimal. Walaupun pada pengujian yang menggunakan dua *web server*, kedua algoritma memberikan hasil yang cukup baik dan *weighted least connection* memiliki hasil 1ms lebih baik pada dua buah pengujian yang menggunakan dua *web server*. Dalam hal mengenai *error request*, kedua algoritma menujukkan hasil yang sangat baik. Akan tetapi, pada pengujian dengan tiga buah *web server* algoritma *weighted least connection* menunjukkan hasil yang lebih baik dibanding dengan algoritma *Weighted Round Robin* dengan selisih yang didapatkan sebesar 25,98%. Pada parameter pengujian penggunaan *memory* atau *memory usage*, sistem yang menonaktifkan *proxy* menghasilkan hasil yang lebih baik dibandingkan pada sistem yang mengaktifkan *proxy* dengan selisih persentase antara keduanya sebesar 60% untuk *single server* dan 61,50% untuk *multiserver*. Algoritma *weighted least connection* mempunyai nilai yang lebih unggul di segala aspek pengujian dibanding algoritma *Weighted Round Robin*.

Meskipun penelitian ini memberikan gambaran yang jelas mengenai perbandingan performa antara algoritma WRR dan WLC, terdapat beberapa keterbatasan yang perlu diperhatikan. Pengujian dilakukan dalam lingkungan virtual menggunakan VirtualBox, yang meskipun dapat merepresentasikan skenario nyata secara umum, tetap memiliki keterbatasan dalam mereplikasi performa dan kompleksitas sistem produksi skala besar. Selain itu, pengujian difokuskan pada trafik HTTP dengan parameter tertentu, tanpa melibatkan jenis trafik lain seperti HTTPS, WebSocket, atau API *burst* yang juga umum digunakan dalam sistem modern. Penelitian ini juga belum mengevaluasi dampak jangka panjang terhadap *resource* seperti konsumsi CPU dan I/O *disk* secara mendetail.

Untuk pengembangan penelitian selanjutnya, disarankan untuk menguji algoritma ini dalam lingkungan *cloud* berbasis kontainer seperti Docker dan Kubernetes, serta menambahkan parameter evaluasi seperti *latency* antar lokasi geografis (*geo-distributed systems*) dan efisiensi algoritma dalam mengelola sumber daya di bawah beban ekstrem. Selain itu, penggabungan algoritma WLC dengan pendekatan berbasis *machine learning* untuk prediksi beban masa depan dapat menjadi arah yang menarik guna menciptakan *sistem load balancing* yang adaptif dan cerdas.

REFERENSI

[1] R. Mustajab, “Pengguna E-Commerce RI Diproyeksi Capai 196,47 Juta pada 2023,” Data Indonesia: Data Indonesia for Better Decision. Valid, Accurate, Relevant, Sep. 04, 2023. Accessed: Dec. 21, 2023. [Online]. Available: https://dataindonesia.id/ekonomi-digital/detail/pengguna-ecommerce-ri-diproyeksi-capai-19647-juta-pada-2023.

[2] E. Raisah &. A. Nugroho, "*Ketahui Apa Itu Load Balancing, Cara Kerja, Kelebihannya, DCloud*. Accessed: Dec. 21, 2023. [Online] Available at: https://dcloud.co.id/blog/apa-itu-load-balancing.html

[3] M. Rosyida, "*Load Balancing : Pengatur Lalu Lintas Data Server,"* Dec. 6, 2023. [Online]. Available: https://www.domainesia.com/berita/load-balancing/

[4] Saputra, I. B., "Perancangan dan Implementasi Load Balancing Web Server menggunakan Haproxy,". *Skripsi Sarjana, Universitas Muhammadiyah Surakarta.* 2012.

[5] Setyawan, A. D., "*Mengenal Apa itu HAProxy serta Konsep Load Balancing, Rumahweb Journal – News, Article, and Tutorial of Web Dev.* Accessed: Dec. 22 2023. [Online]. Available at: https://www.rumahweb.com/journal/haproxy-adalah/

[6] Alsyaibani & Omar, M. A., "Performa Algoritma Load Balance pada Server Web Apache dan Nginx dengan Database Postgresgl," *Skripsi Sarjana, Universitas Negeri Yogyakarta,* 2013.

[7] Sumbayak, G. S. N. H. &. P. R., "Implementasi Algoritma Weighted Least Connection Berbasis Agen Pada POX Controller Untuk Load Balancing Web Server Pada Software Defined Network.". *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer.* 2019. Hal 7335-7344.

[8] Supramana, &. P. I. G. L. P, "Implementasi Load Balancing pada Web Server dengan Menggunakan Apache". *Jurnal Manajemen Informatika.* 2016.

[9] Anicas, M., "*Introduction to HAProxy and Load Balancing Concept,"* Accessed: Feb. 15, 2024. [Online]. https://www.digitalocean.com/community/tutorials/an-introduction-to-haproxy-and-load-balancing-concepts.

[10] Bustomi, Z. S. M. A. M. I. &. H. K. F. H., n.d, "Load Balancing Web Server Menggunakan Nginx pada Lingkungan Virtual". *Jurnal Informatika: Jurnal pengembangan IT,* 2019.

[11] Rachmawan, D. I. D. &. A. H., "Penerapan Teknik Load Balancing pada Web Server Lokal dengan Metode NTH Menggunakan Mikrotik," *Jurnal Penelitian Ilmu Komputer, System Embedded & Logic,* pp. 98-108, 2016.

[12] Rahmatulloh, A. &. M. F., "Implementasi Load Balancing Web Server menggunakan Haproxy dan Sinkronisasi File pada Sistem Informasi Akademik Universitas Siliwangi," *Jurnal Nasional Teknologi dan Sistem Informasi.* 2017

[13] Apriliansyah, F. F. I. &. I. A., "Implementasi Load Balancing Pada Web Server Menggunakan Nginx," *Jurnal Teknologi dan Manajemen Informatika,* 2020.

[14] Riska, &. A. H., "Analisa Dan Perancangan Load Balancing Web Server menggunakan HAProxy. Techno.COM". 2021 hal. 552-565.

[15] Riskiono, S. D. &. P. D., "Analisis Perbandingan Server Load Balancing dengan Haproxy & Nginx dalam Mendukung Kinerja Server E- Learning,". *Jurnal Telekomunikasi dan Komputer. 2020.*

[16] Singh, G. &. K. K., "An Improved Weighted Least Connection Scheduling Algorithm for Load Balancing in Web Cluster Systems,". *International Research Journal of Engineering and Technology.* 2018.

[17] Oliver, H., "*Load Balancing Matchup: Round Robin vs. Weighted Round Robin, Best DNS Servers - Uptime & Performance''* Accessed: Apr. 22, 2024. [Online] https://constellix.com/news/load-balancing-round-robin-vs-weighted-round-robin.

[18] Johansson, A, "HTTP Load Balancing Performance Evaluation of HAProxy, NGINX, Traefik and Envoy with the Round-Robin Algorithm," *Thesis of Undergraduate, University of Skovde).2022*

[19] Mbarek, F. &. M. V, "Load Balancing Based on Optimization Algorithms: An Overview," *Journal of Telecommunications and Information Technology.* 2019.