

Analisis Kinerja Load Balancing Webserver Menggunakan Haproxy Terintegrasi Dengan Grafana Sebagai Monitoring Dan Notifikasi Telegram

Elsan Pentanugraha¹⁾ Agus Sehatman Saragih²⁾ Efrans Christian³⁾

¹⁾Jurusan Teknik Informatika Fakultas Teknik Universitas Palangka Raya
Kampus Tunjung Nyaho, Jl. Yos Sudarso 73112

¹⁾elsanpentanugraha687@gmail.com

ABSTRAK

Di era digital saat ini, *Website* sangat penting bagi organisasi atau perusahaan. Namun, semakin banyak pengunjung pada *Website*, semakin besar beban *server*. Ini dapat membuat *server* menjadi lambat atau *down*. Solusi untuk masalah ini adalah *load balancing*. *Haproxy* adalah perangkat lunak *open-source* yang bisa digunakan sebagai *load Balancer*. *Monitoring* platform yang digunakan dalam penelitian ini adalah *Grafana* yang memantau sumber daya *server* secara *real-Time* dan menyediakan visualisasi data yang mudah dipahami.

Tujuan penelitian ini adalah menganalisis kinerja pada *load balancing Web server*, memberikan gambaran kinerja *Haproxy* sebagai *load balancing* dalam menangani beban berat pada *Web server*, dan memberikan solusi alternatif untuk meningkatkan kinerja *Web server*. Tahapan dalam penelitian ini peneliti melakukan analisis kebutuhan, perancangan topologi, implementasi, dan pengujian *Web server* menggunakan *Apache JMeter*. Penelitian ini menggunakan *Apache JMeter* untuk menguji keandalan dan stabilitas *Web server* saat menangani lonjakan permintaan yang besar pada skenario *single server* dan *Load balancing*. Peneliti melakukan pengujian kinerja *load balancing Web server* dengan beberapa parameter yaitu *throughput*, *Response Time*, *error rate*, *memory utilization*, dan *CPU utilization*.

Berdasarkan analisis dan penelitian yang dilakukan tentang kinerja *load balancing* pada *Web server* menggunakan *HAProxy* dan integrasi dengan *Grafana* sebagai *Monitoring* serta notifikasi *Telegram*, dapat disimpulkan bahwa *Grafana* yang terintegrasi dengan *HAProxy* efektif sebagai alat *Monitoring* performa *Web server* yang memberikan visualisasi data yang jelas dan mudah dipahami. Hal ini memudahkan proses analisis dan evaluasi performa *Web server*. Penggunaan *load balancing* juga meningkatkan ketersediaan *server* dalam menangani permintaan, terbukti dengan penurunan persentase kegagalan *server* secara drastis.

Kata Kunci : *Load Balancing, HAProxy, Analisis Kinerja, Web server.*

Abstract

In the current digital era, websites are crucial for organizations or companies. However, as the number of visitors to a website increases, the server's burden also grows larger. This can lead to server slowness or even downtime. The solution to this problem is load balancing. Haproxy is an open-source software that can be used as a load balancer. The monitoring platform used in this research is Grafana, which monitors server resources in real-time and provides easily understandable data visualization.

The objectives of this research are to analyze the performance of load balancing on web servers, provide an overview of Haproxy's performance as a load balancer in handling heavy loads on web servers, and offer alternative solutions to improve web server performance. The stages in this research involve analyzing requirements, designing topology, implementation, and testing the web server using Apache JMeter. Apache JMeter is used to test the reliability and

stability of the web server when handling large request surges in both single server and load balancing scenarios. The performance of the load-balanced web server is evaluated based on several parameters, including throughput, response time, error rate, memory utilization, and CPU utilization.

Based on the analysis and research conducted on the performance of load balancing on web servers using HAProxy and integrated with Grafana for monitoring and Telegram notifications, it can be concluded that Grafana integrated with HAProxy is an effective tool for monitoring web server performance, providing clear and easily understandable data visualization. This facilitates the process of analyzing and evaluating web server performance. The use of load balancing also improves server availability in handling requests, as evidenced by a significant decrease in server failure percentage.

Keywords : Load Balancing, HAProxy, Performance Analisis, Web Server.

1. PENDAHULUAN

Di era digital saat ini, *Website* menjadi salah satu hal yang sangat penting bagi suatu organisasi atau perusahaan. *Website* digunakan sebagai sarana promosi, informasi, dan transaksi. Namun, dengan semakin banyaknya pengunjung pada suatu *Website*, maka semakin besar pula beban yang harus ditangani oleh *server*. Hal ini dapat mengakibatkan *server* menjadi lambat atau bahkan *down*, yang akan berdampak pada kualitas layanan dan citra perusahaan. Permasalahan pada umumnya tidak jarang pengelola *server* kurang mengelola dengan baik *Web server*. Oleh sebab itu, kegagalan dan kelebihan beban pada *server* sering terjadi karena banyak pengguna menggunakannya secara bersamaan dikarenakan hanya menggunakan satu *server* untuk menerima banyak permintaan dan tidak ada *server* lain yang membantu ketika terjadi *overload* [1].

Untuk mengatasi masalah ini, salah satu solusi yang dapat digunakan adalah *load balancing*. *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 [2]. Dalam hal ini, *Haproxy* merupakan salah satu perangkat lunak *open-source* yang dapat digunakan sebagai *load Balancer*.

Namun, hanya menggunakan *load Balancer* saja tidak cukup. Perlu dilakukan *Monitoring* performa *Web server* secara *real-Time* untuk mengetahui kondisi *server* dan mengambil tindakan yang diperlukan jika terjadi masalah. *Platform Monitoring* yang digunakan dalam penelitian ini adalah aplikasi pihak ketiga yaitu *Grafana*, yang memantau sumber daya *server* secara *real Time* dan menyediakan visualisasi data yang mudah dipahami. *Grafana* adalah perangkat lunak visualisasi dan analitik yang bersifat *opensource*. *Grafana* memungkinkan untuk memvisualisasikan, mengingatkan, dan menjelajahi metrik disimpan [3]. Selain itu, dengan memanfaatkan *Telegram* sebagai media notifikasi, administrator dapat menerima notifikasi secara cepat dan efisien jika terjadi masalah pada *server*.

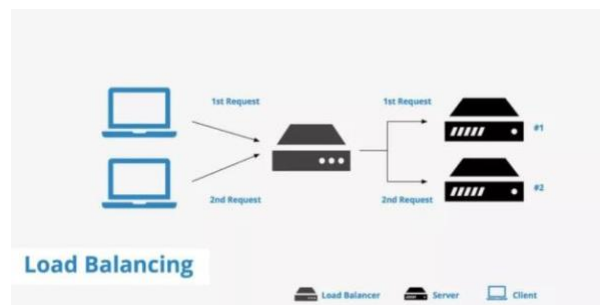
Penelitian ini bertujuan untuk menguji kinerja *Haproxy* sebagai *Web server load Balancer*, menggunakan *Grafana* sebagai alat *Monitoring* untuk mendapatkan informasi secara *realTime* mengenai kinerja *Web server* dan notifikasi *Telegram* ketika terjadi kesalahan pada *server*. Selain itu, penelitian ini menguji sistem menggunakan alat *Stress testing* untuk menguji keandalan dan stabilitas *Web server* dan *load Balancer* saat menangani lonjakan permintaan yang besar. Hasil penelitian ini bertujuan untuk memberikan gambaran kinerja *Haproxy* sebagai *load Balancer* dalam menangani beban berat pada *Web server* dan memberikan solusi alternatif untuk meningkatkan kinerja *Web server*.

2. TINJAUAN PUSTAKA

a. *Load Balancing*

Load balancing adalah teknik untuk mendistribusikan beban lalu lintas secara merata di dua atau lebih batang sehingga lalu lintas mengalir secara optimal, memaksimalkan *throughput*, meminimalkan waktu respons, dan menghindari kelebihan beban pada salah satu jalur koneksi [8].

Load Balancing juga membantu menghindari terjadinya *downTime* pada *server* dengan memastikan bahwa lalu lintas yang datang didistribusikan dengan merata ke seluruh *server* yang tersedia. Hal ini memungkinkan terjadinya manajemen beban kerja yang lebih baik dan mengurangi kemungkinan *server* mengalami kegagalan atau kerusakan. *load Balancer* akan memindahkan lalu lintas ke *server* lain yang masih bisa menangani permintaan. Gambar 1 menunjukkan contoh topologi dari *load balancing Web server*.



Gambar 1 Topologi *Load Balancing* [3]

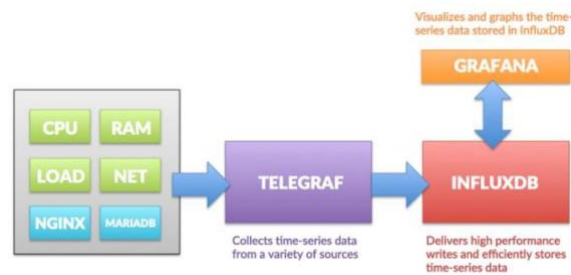
Dalam *load balancing Web server*, terdapat beberapa algoritma yang dapat digunakan untuk membagi beban kerja antara beberapa *server Web* yang tersedia. Salah satunya adalah *Round Robin*. Algoritma *Round Robin* adalah algoritma *load balancing* paling dasar yang tersedia pada *HAProxy*. Algoritma ini secara bergantian memilih *server backend* yang tersedia untuk menerima permintaan. Setelah permintaan terkirim ke *server backend* pertama, permintaan selanjutnya akan dikirim ke *server backend* kedua, dan seterusnya. Setelah semua *server backend* menerima permintaan, permintaan akan kembali lagi ke *server backend* pertama.

b. *Haproxy*

HAProxy adalah produk *open source* yang digunakan untuk membangun sistem *load balancing* dan *failover* untuk aplikasi berbasis *TCP* dan *HTTP*. *Software* ini sangat cocok untuk *Website* yang memiliki *traffic* harian yang tinggi. [4]. *Haproxy* juga memungkinkan pengguna untuk mengatur distribusi lalu lintas ke *server* dengan beberapa algoritma *load balancing*, di antaranya adalah *Round Robin*, *Least Connections*, *IP Hashing*, dan lain sebagainya.

c. *Monitoring*

Monitoring adalah proses pemantauan dan pengukuran sistem atau proses secara terus-menerus untuk memastikan bahwa mereka berfungsi dengan baik dan sesuai dengan tujuan yang diinginkan. Tujuan utama dari *Monitoring* adalah untuk mengidentifikasi masalah atau anomali yang mungkin terjadi, sehingga dapat diambil tindakan yang tepat untuk mengatasi masalah tersebut sebelum mereka mempengaruhi kinerja atau ketersediaan sistem. *Monitoring* dapat dilakukan pada berbagai tingkatan, mulai dari level perangkat keras (seperti *CPU*, memori, dan *disk*), jaringan (seperti lalu lintas jaringan dan latensi), hingga aplikasi (seperti waktu respons, jumlah permintaan, dan kesalahan *server*).



Gambar 2. Alur Monitoring dengan Telegraf, InfluxDB, dan Grafana [6].

d. **Telegraf**

Telegraf merupakan salah satu jenis *daemon* yang dapat berjalan diberbagai macam jenis *server* manapun, serta dapat mengumpulkan berbagai macam jenis informasi metrik dari sistem *server* (*CPU*, memori, storage dan lainnya). *Telegraf* juga memiliki berbagai macam jenis *plugin output* yang berfungsi untuk mengirim metrik ke berbagai jenis *datastore* termasuk *InfluxDB*, *Graphite*, *OpenTSDB* serta *Datalog* [7]. Cara kerja *Telegraf* dalam mengumpulkan data dari *server* yaitu dengan menggunakan *plugin* untuk mengumpulkan data dari berbagai sumber, seperti sistem operasi, aplikasi, dan basis data. *Plugin* tersebut dapat disesuaikan dan dikonfigurasi sesuai dengan kebutuhan pengguna. Setelah *plugin* dikonfigurasi, *Telegraf* mulai mengumpulkan data metrik dari *server* dan sumber lainnya.

e. **InfluxDB**

InfluxDB adalah sebuah *database Time-series open source* yang dikembangkan oleh *InfluxData*. *InfluxDB* dibuat dalam bahasa *GO* dan berfungsi sebagai media penyimpanan untuk metrik, data sensor, dan analisis secara *real-Time* [7]. Cara kerja *InfluxDB* dalam menyimpan dan mengelola data *Time-series* adalah dengan menyimpan data dalam bentuk *Time-series*, yaitu data yang berhubungan dengan waktu. *InfluxDB* dapat diintegrasikan dengan berbagai aplikasi *Monitoring* atau analitik, seperti *Grafana* atau *Kapacitor*, sehingga memungkinkan pengguna untuk melakukan visualisasi data dan melakukan pemrosesan data lebih lanjut.

f. **Grafana**

Grafana adalah perangkat lunak *open source* yang membaca matriks data untuk mengubahnya menjadi grafik atau data tertulis. *Grafana* sering digunakan untuk analisis dan *Monitoring* [8]. Selain itu, *Grafana* juga memiliki kemampuan untuk memungkinkan kustomisasi yang tinggi, dan memungkinkan integrasi dengan berbagai layanan dan sistem lainnya seperti *Prometheus*, *InfluxDB*, *Elasticsearch*, dan banyak lagi. *Grafana* sangat populer di kalangan profesional IT dan Data Scientist untuk memantau kinerja sistem, visualisasi data, dan analitik.

g. **Telegram**

Telegram merupakan sebuah aplikasi pesan instan berbasis *cloud* yang berfokus pada kecepatan dan keamanan [9]. *Telegram* sebagai salah satu media notifikasi yang digunakan untuk memberikan pemberitahuan jika terjadi permasalahan pada *server* dengan memanfaatkan *bot*. *Telegram Bot* adalah aplikasi pihak ketiga yang berjalan di sistem *Telegram* dan merupakan salah satu layanan yang disediakan oleh aplikasi *Telegram* [10].

h. Apache JMeter

Apache JMeter adalah salah satu Tools open-source yang digunakan untuk pengujian kinerja aplikasi dan pengujian beban (load testing) pada aplikasi Web atau protokol lainnya. Tools ini dikembangkan oleh Nginx Software Foundation dan ditulis dalam bahasa pemrograman Java. Tools ini juga mendukung pengujian kinerja pada berbagai teknologi seperti server aplikasi, database, dan messaging middleware. Apache JMeter memiliki antarmuka pengguna yang intuitif dan dapat dioperasikan oleh pengguna yang memiliki pengetahuan dasar tentang pengujian kinerja.

i. Parameter pengujian Kinerja

Performa load balancing Webserver dapat diukur menggunakan beberapa parameter, di antaranya adalah CPU Utilization, Memory Utilization, Response Time, throughput, dan error rate.

1) CPU utilization

CPU utilization (penggunaan CPU) adalah ukuran seberapa banyak CPU pada sebuah sistem digunakan pada suatu waktu tertentu. Satuan yang digunakan untuk penggunaan CPU yaitu % (Persen).

2) Memory utilization

Memory utilization atau penggunaan memori adalah ukuran seberapa banyak memori pada sebuah sistem digunakan pada suatu waktu tertentu. Pada Penggunaan memori, satuan yang digunakan adalah % (Persen).

3) Response Time

Response Time adalah waktu yang dibutuhkan oleh server untuk memproses permintaan (request) dari client dan mengirimkan respon (Response) kembali ke client. Response Time yang rendah menunjukkan kinerja server yang baik. Satuan umum yang digunakan untuk mengukur Response Time adalah milidetik (ms) atau detik (s).

4) Throughput

Throughput adalah nilai rata-rata pengiriman yang dapat diproses oleh server dalam satu waktu tertentu yang diukur dalam satuan bit per second (bps atau bit/s). Semakin tinggi throughput, semakin baik kinerja server. Rumus menghitung throughput ditunjukkan pada persamaan berikut.

$$\text{Throughput} = \frac{\text{Jumlah packet data yang dikirim (bit)}}{\text{Waktu pengiriman paket (second)}}$$

Pengelompokan standarisasi parameter throughput ditunjukkan pada tabel berikut.

Tabel 1 Kategori nilai Throughput

Kategori Througput	Throughput (bit/s)	Indeks
Sangat Bagus	>2,1 Mbps	4
Bagus	1200 Kbps – 2,1 Mbps	3
Cukup	700 – 1200 kbps	2
Kurang Bagus	338 – 700 kbps	1
Buruk	0 - 338 kbps	0

5) Error Rate

Error Rate adalah jumlah permintaan (request) yang gagal diproses dibandingkan dengan jumlah total permintaan. Error rate yang rendah menunjukkan kinerja server yang baik.

Satuan yang digunakan untuk *error rate* yaitu % (Persen).

3. METODE PENELITIAN

a. Tahapan Penelitian

Adapun Tahapan penelitian yang digunakan dalam penelitian ini adalah sebagai berikut:



Gambar 3 Tahapan Penelitian

b. Analisis Kebutuhan

- Kebutuhan Perangkat Keras (Hardware)

Perangkat keras yang digunakan dalam penelitian ini adalah 1 buah laptop yang digunakan sebagai *client* dan perangkat yang digunakan penulis untuk melakukan penelitian ini dengan spesifikasi pada Tabel 3.1.

Tabel 2 Spesifikasi Perangkat Keras

OS	Windows 11 Home Single 22H2
Processor	12th Gen Intel Core i7-12650H 2.30 GHz
RAM	16 GB
SSD	512 GB

- Kebutuhan Perangkat Lunak (Software)

Pada penelitian ini dibangun 4 mesin *Virtual* pada *VMware Workstation* yaitu 1 *Load Balancer* dengan *HAProxy* dan juga *Grafana* sebagai *Monitoring* dan 3 *Web Server* dengan *Nginx*. Spesifikasi perangkat *Virtual* ini tercantum dalam Tabel 3.2.

Tabel 3 Spesifikasi Perangkat Virtual

<i>Virtual Machine</i>	Spesifikasi	
Load Balancer	OS	<i>Debian 11.6 Bullseye</i>
	RAM	4 GB
	Disk	80 GB
	Nama VM	<i>Load Balancer</i>
Web Server 1	OS	<i>Debian 11.6 Bullseye</i>
	RAM	1 GB
	Disk	50 GB
	Nama VM	<i>Nginx Web Server 1</i>
Web Server 2	OS	<i>Debian 11.6 Bullseye</i>
	RAM	2 GB
	Disk	50 GB

Web Server 3	Nama VM	<i>Nginx Web Server 2</i>
	OS	<i>Debian 11.6 Bullseye</i>
	RAM	2 GB
	Disk	50 GB

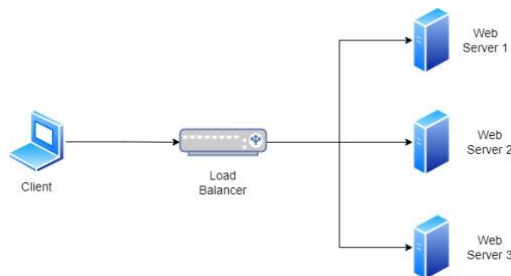
Perangkat lunak yang digunakan sebagai alat dan aplikasi pada penelitian ini disajikan pada Tabel 3.3.

Tabel 4 Alat dan Aplikasi

No	Software	Versi	Fungsi
1	Drawio	21.2.9	Desain Topologi perangkat
2	VMware Workstation	17.0	Virtual Enviroment
3	Nginx	1.18.0	Web Server
4	Telegraf	1.26.1	Agent yang mengumpulkan data resource perangkat
5	InfluxDB	1.8.10	Database yang menyimpan data dari Telegraf secara Time-series
6	Grafana	9.4.7	Visualisasi resource perangkat
7	Telegram	9.6.5	Notifikasi Ketika terjadi masalah
8	Apache JMeter	5.4.3	Alat Pengujian Web Server
9	Browser Chrome	113.0	Menampilkan Halaman Web

c. Perancangan Topologi

Adapun topologi load balancing Web server yang dirancang pada penelitian ini disajikan pada gambar 3.2.



Gambar 4 Desain Topologi Load Balancing Web Server

Dari Gambar 3.2 diatas, *Virtual Machine* yang akan digunakan berjumlah 4 *Virtual Machine server*. 1 *Virtual Machine* akan dikonfigurasi sebagai *load Balancer* dengan menggunakan *HAProxy* sebagai *Tools load balancing* dan juga *server Monitoring* menggunakan *Telegraf* sebagai *agent* pengumpul data matriks *server*, *InfluxDB* yang digunakan sebagai *database* penyimpanan data matriks yang dikirimkan oleh *Telegraf* secara *Time-series*, dan *Grafana* sebagai *Tool* untuk memvisualisasikan data matriks tersebut kedalam sebuah bentuk grafik, 3 *Virtual Machine server* lainnya akan dikonfigurasi sebagai *Web server* menggunakan *Nginx* sebagai layanan *Web server*.

d. Skenario Pengujian

Proses pengujian bertujuan untuk mendapatkan hasil dari kinerja dari single server dan juga *load balancing Web server*. *Tools* atau aplikasi yang digunakan pada proses pengujian ini adalah *Apache JMeter* untuk mendapatkan nilai parameter *Throughput*, *Response Time*, dan *error rate*. Untuk mendapatkan nilai parameter *CPU Utilization* dan

Memory Utilization, *Tools* yang digunakan adalah *Grafana* yang telah terintegrasi pada *load Balancer*.

Untuk mendapatkan nilai seperti *CPU Utilization*, *Memory Utilization*, *throughput*, *Response Time*, dan *error rate* penelitian dilakukan dengan beberapa percobaan melalui skenario pengujian dengan jumlah koneksi yang berbeda-beda pada tingkatan pengujian menengah (100 hingga 500 *thread*) dan pada tingkatan pengujian Tinggi (1000 hingga 5000 *thread*). Maka ditentukan jumlah *thread* yang akan digunakan yaitu 100, 200, 500, 1000, dan 2000 koneksi. *Request rate* rata-rata setiap koneksi adalah sebesar 100 *request* per detik dimana pada tiap tingkatan *thread*, beban kerja akan digrup dengan jumlah 100 *thread* di tiap grup. Pengujian dilakukan berulang sebanyak 10 kali untuk mendapatkan hasil yang akurat serta mengurangi resiko kesalahan dalam pengujian. Pada tabel berikut adalah skenario dari tingkatan pengujian kinerja sistem.

Tabel 5 Skenario Tingkatan Pengujian

Jumlah Koneksi (<i>Thread</i>)	<i>Request per detik</i>
100	100
200	100
500	100
1000	100
2000	100

3.6.1 Pengujian

Tahapan Pengujian yang dilakukan untuk mendapatkan nilai *Throughput*, *Response Time*, *error rate*, *CPU Utilization*, dan *Memory Utilization* dari kinerja *Single Web server* dan juga kinerja *load balancing web server* menggunakan aplikasi *Apache JMeter* dan *Grafana* adalah sebagai berikut.

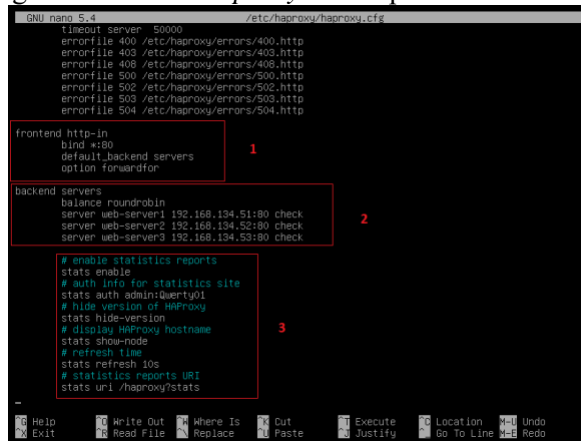
- 1) Menjalankan *Web server* yang akan diuji dan juga menjalankan *server Monitoring*.
- 2) Mengakses *Dashboard Grafana* dengan mengetikkan alamat *IP Address* dan juga *port* yang digunakan.
- 3) Melakukan Konfigurasi pengujian pada aplikasi *Apache JMeter* dengan jumlah koneksi 100, 200, 500, 1000, dan 2000 koneksi dan *Request rate* yaitu 100 *request* per detik.
- 4) Pengujian dilakukan sebanyak 10 kali untuk setiap skenario pengujian.
- 5) Pengambilan data dilakukan setiap pengujian selesai. Setelah data didapatkan selanjutnya akan dilakukan analisis data *throughput*, *Response Time*, *error rate*, *CPU Utilization*, dan *Memory Utilization*.

4. HASIL DAN PEMBAHASAN

a. Konfigurasi

Pada Gambar 5 konfigurasi *haproxy*. Point pertama pada konfigurasi tersebut adalah inisialisasi dari konfigurasi *frontend* yang menerima lalu lintas *HTTP* dengan melakukan *binding* ke *port* 80 dan mendistribusikannya ke *backend server* yang tersedia. Point kedua, adalah konfigurasi untuk menginisialisasikan *server backend*. Baris “*balance roundrobin*” merupakan metode *load balancing* yang akan mengarahkan setiap permintaan klien yang dilakukan secara bergantian ke setiap *server backend* yang tersedia sesuai dengan urutan *server backend* yang didefinisikan dalam konfigurasi. Pada baris “*server*” merupakan *server-server backend* yang diinisialisasikan sebagai *server* yang akan menerima lalu lintas. Definisi *server* pada baris ini dilakukan dengan menuliskan alamat IP dan *port server* yang akan menerima permintaan dan “*check*” merupakan konfigurasi agar *haproxy* secara teratur memeriksa status *server* untuk memastikan ketersediaan dan menghindari mengirimkan permintaan ke *server* yang tidak aktif. Dan point yang ketiga, merupakan konfigurasi untuk

melakukan *enable* pemantauan statistik dan status *server backend* yang tersedia dengan mengetikkan alamat ip dengan tambahan “/haproxy?stats” pada alamat.



```
GNU nano 3.1 /etc/haproxy/haproxy.cfg
# timeout server 50000
errorfile 400 /etc/haproxy/errors/400.http
errorfile 408 /etc/haproxy/errors/408.http
errorfile 409 /etc/haproxy/errors/409.http
errorfile 500 /etc/haproxy/errors/500.http
errorfile 502 /etc/haproxy/errors/502.http
errorfile 503 /etc/haproxy/errors/503.http
errorfile 504 /etc/haproxy/errors/504.http

frontend http-in
  bind *80
  default_backend servers
  option forwardfor

backend servers
  balance roundrobin
  server web-server1 192.168.134.51:80 check
  server web-server2 192.168.134.52:80 check
  server web-server3 192.168.134.53:80 check

# enable statistics reports
stats enable
# auth info for statistics site
stats auth admin:qwerty01
# hide version of HAProxy
stats hide-version
# display HAProxy hostname
stats show-node
# refresh time
stats refresh 10s
# statistics reports URI
stats uri /haproxy?stats
```

Gambar 5 Konfigurasi HAProxy

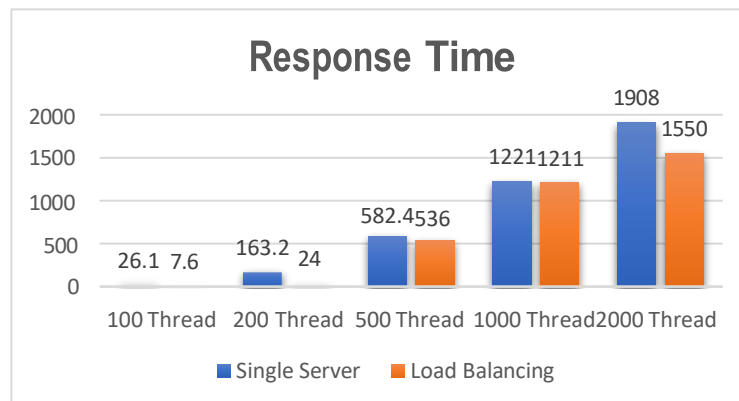
b. Profil Beban Kerja

Profil beban kerja menggambarkan pola dan tingkat permintaan yang diberikan kepada sistem selama pengujian.

- i. Jumlah Klien: Pengujian dilakukan menggunakan beberapa jumlah klien yang meminta secara bersamaan yaitu pada tingkatan menengah menggunakan 100, 200, dan 500 Threads, kemudian pada tingkatan tinggi menggunakan 1000 dan 2000 Threads.
- ii. Jenis Permintaan: Permintaan yang dilakukan oleh klien adalah permintaan HTTP yaitu GET untuk mengakses halaman utama Web server dan POST untuk melakukan input data kedalam database.
- iii. Kepadatan Permintaan: Permintaan dilakukan dengan kepadatan tinggi, yaitu setiap 1 detik, setiap klien membuat permintaan.
- iv. Request rate: Request rate atau tingkat permintaan merujuk pada jumlah permintaan yang dikirimkan ke Webserver dalam satu unit waktu tertentu. Pengujian dilakukan dengan menggunakan 100 Request per detik.

c. Analisis Response Time

Berdasarkan data pada Gambar 6, bisa dilihat jika pada skenario tanpa *load balancing*, waktu respons meningkat secara signifikan seiring dengan kenaikan jumlah *thread*. Hal ini menunjukkan jika semakin banyak permintaan yang diproses, semakin lama waktu respons yang diperlukan.



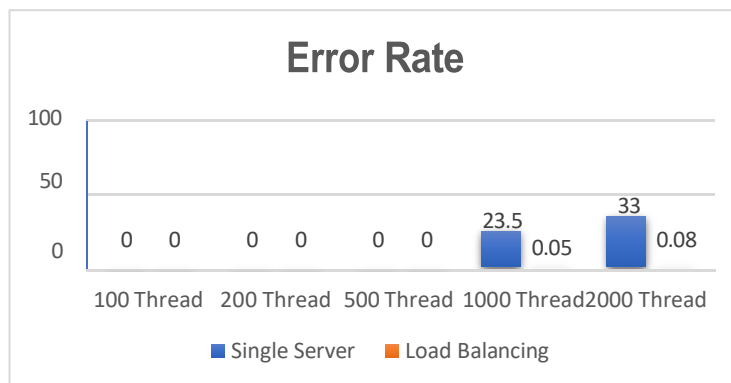
Gambar 6 Grafik Data Pengujian Response Time

Pada skenario dengan *load balancing*, waktu respons cenderung lebih rendah dibanding

dengan skenario tanpa *load balancing* pada tiap tingkatan *thread*. *Load balancing* membantu mendistribusikan beban kerja secara merata di antara *server-server* yang tersedia, sehingga mengurangi waktu respons secara keseluruhan.

Walaupun demikian, pada *thread* 1000 dan 2000, waktu respons pada skenario dengan *load balancing* tidak mengalami penyusutan yang signifikan dibanding dengan skenario tanpa *load balancing*. Hal ini bisa jadi diakibatkan oleh batasan sumber daya *server* ataupun aspek lain yang mempengaruhi kinerja sistem pada tingkatan beban kerja yang besar.

d. Analisis Error Rate



Gambar 7 Grafik Data Pengujian Error Rate

Dari data tersebut, bisa dilihat jika pada skenario tanpa *load balancing*, ketika beban kerja (jumlah *thread*) mencapai 1000 serta 2000, terjadi kenaikan yang signifikan dalam tingkatan kesalahan. Pada *thread* 1000, tingkatan kesalahan mencapai 23,5%, sementara itu pada *thread* 2000, tingkatan kesalahan meningkat jadi 33,0%. Hal ini menunjukkan jika sistem tanpa *load balancing* mungkin mengalami kinerja yang kurang baik ataupun ketidakseimbangan pada saat mengalami beban kerja yang besar yang dapat menyebabkan peningkatan tingkatan kesalahan.

Di sisi lain, pada skenario dengan *load balancing*, tingkatan kesalahan tetap rendah ataupun bahkan mendekati nol pada seluruh *thread* yang diamati. Walaupun pada *thread* 1000 serta 2000 terdapat tingkat kesalahan yang tercatat, tingkatan tersebut sangat kecil, ialah 0,05% serta 0,08% masing-masing. Hal ini menunjukkan jika pemanfaatan *load balancing* membantu dalam mempertahankan kinerja sistem yang stabil serta mengurangi resiko terjadinya kesalahan sehingga dapat meningkatkan ketersediaan dan keandalan sistem secara keseluruhan.

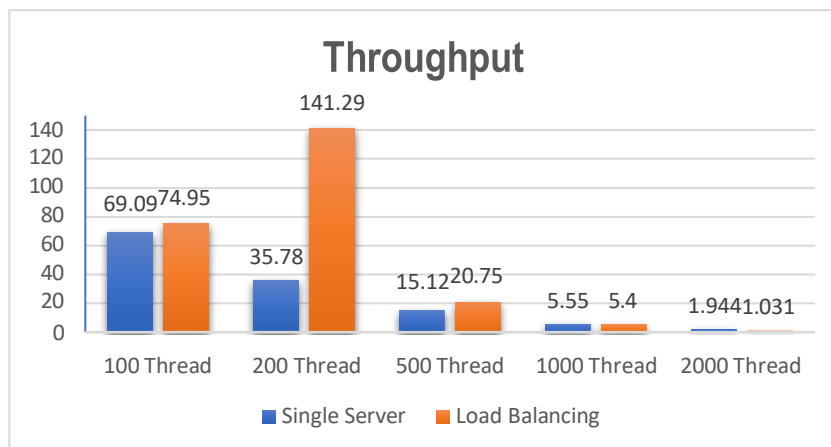
e. Analisis Throughput

Berdasarkan data yang ada, terlihat bahwa *throughput* di dalam skenario tanpa penggunaan *load balancing* rata-rata menyusut seiring dengan penambahan jumlah *thread*. Ini mengindikasikan jika terdapat limitasi dalam kapabilitas sistem untuk melakukan transfer data dengan kecepatan yang tetap disaat beban kerja meningkat.

Akan tetapi, dalam skenario dimana *load balancing* digunakan, *throughput* rata-rata lebih unggul dibanding dengan skenario tanpa penggunaan *load balancing* pada tiap tingkat *thread*. Pemanfaatan *load balancing* membantu dalam meratakan distribusi beban kerja antara *server-server* yang ada, yang pada kesimpulannya dapat meningkatkan *throughput* secara total.

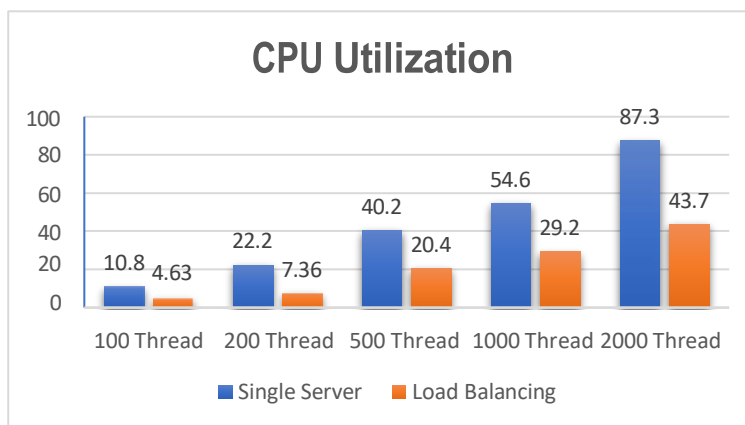
Meski demikian, *throughput* dalam skenario yang menggunakan *load balancing* umumnya lebih baik, tetapi pada *thread* 2000 *throughput* menampilkan penurunan dibanding dengan skenario tanpa penggunaan *load balancing*. Ini kemungkinan besar diakibatkan oleh keterbatasan sumber daya atau pengaturan *load balancing* yang kurang efisien untuk jumlah

thread tersebut.



Gambar 8 Grafik Data Pengujian Throughput

f. Analisis CPU Utilization



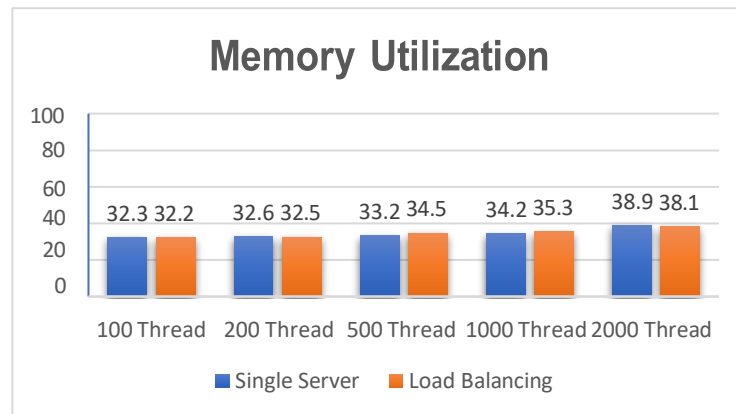
Gambar 9 Grafik Data Pengujian CPU Utilization

Dari data tersebut, dapat dilihat jika pada skenario tanpa *load balancing*, penggunaan *CPU* bertambah seiring dengan kenaikan jumlah *thread*. Hal ini menunjukkan jika semakin banyak permintaan yang diproses, semakin besar penggunaan *CPU* yang diperlukan. Pada *thread* 2000, penggunaan *CPU* mencapai 87,3%, yang menunjukkan beban kerja yang sangat besar pada *server*.

Pada skenario dengan *load balancing*, penggunaan *CPU* lebih menyeluruh di antara *server-server* yang ada. Penggunaan *CPU* pada tiap *server* cenderung lebih rendah dibanding dengan skenario tanpa *load balancing*. Hal ini menunjukkan jika dengan memanfaatkan *load balancing*, beban kerja dapat didistribusikan dengan lebih merata di antara *server-server*, mengurangi beban *CPU* pada tiap *server*.

Walaupun penggunaan *CPU* pada skenario dengan *load balancing* masih bertambah seiring dengan kenaikan jumlah *thread*, peningkatannya tidak sebesar pada skenario tanpa *load balancing*. Hal ini menunjukkan efektivitas *load balancing* dalam mengurangi beban *CPU* serta mempertahankan penggunaan *CPU* yang lebih seimbang di antara *server-server*.

g. Analisis Memory Utilization



Gambar 10 Grafik Data Pengujian Memory Utilization

Dari data tersebut, bisa dilihat jika pada skenario tanpa *load balancing*, penggunaan memori meningkat bersamaan dengan kenaikan jumlah *thread*. Hal ini menunjukkan jika semakin banyak permintaan yang diproses, semakin banyak memori yang diperlukan.

Pada skenario dengan *load balancing*, penggunaan memori pula senantiasa meningkat seiring dengan kenaikan jumlah *thread*, akan tetapi dengan perbandingan penggunaan memori yang lebih seimbang di antara *server-server*. Penggunaan memori pada tiap *server* cenderung lebih rendah dibanding dengan skenario tanpa *load balancing*. Hal ini menunjukkan kalau dengan menggunakan *load balancing*, beban memori bisa didistribusikan dengan lebih menyeluruh di antara *server-server*, mengurangi beban memori pada tiap *server*.

Walaupun penggunaan memori pada skenario dengan *load balancing* masih meningkat seiring dengan kenaikan jumlah *thread*, peningkatannya tidak sebesar pada skenario tanpa *load balancing*. Hal ini menampilkan efektivitas *load balancing* dalam mengurangi beban memori serta mempertahankan penggunaan memori yang lebih seimbang di antara *server-server*.

5. KESIMPULAN

a. Kesimpulan

Berdasarkan analisis dan penelitian yang telah dilakukan terkait dengan kinerja *load balancing* pada *Web server* menggunakan *HAProxy* dan integrasi dengan *Grafana* sebagai *Monitoring* serta notifikasi *Telegram*, dapat disimpulkan Implementasi *load balancing* menggunakan *HAProxy* dapat meningkatkan kinerja *server Website*. Dalam skenario dengan *load balancing*, *Response Time* lebih rendah dibandingkan dengan *single server*. Selain itu, penggunaan *load balancing* juga meningkatkan ketersediaan *server* dalam menangani permintaan hal ini dibuktikan dengan persentasi kegagalan *server* pada skenario *load balancing* hanya 0,08% sehingga dalam menangani permintaan, kegagalan pada *load balancing* berkurang secara drastis dibandingkan dengan skenario *single server* yang mencapai 33,0%.

Dari sisi sumber daya *server*, penggunaan *CPU* dengan beban 2000 *thread* pada skenario *load balancing* hanya 43,7%. Penggunaan *CPU* pada skenario ini berkurang secara signifikan dibanding dengan *single server* yang mencapai 87,3% tetapi dari sisi penggunaan memori penurunan penggunaannya tidak signifikan.

Pada *throughput* juga mengalami peningkatan jumlah data yang dikirimkan pada tingkatan beban tertentu. Namun, pada *thread* yang mencapai 1000 dan 2000 *thread* dalam penelitian ini, *throughput* justru menurun. Hal ini mungkin disebabkan oleh keterbatasan

sumber daya atau pengaturan *load balancing* yang tidak optimal untuk jumlah *thread* tersebut. *Grafana* yang terintegrasi dengan *HAProxy* efektif digunakan sebagai alat *Monitoring* performa *Webserver* yang memberikan visualisasi data yang jelas dan mudah dipahami sehingga memudahkan dalam proses analisis dan evaluasi performa *Webserver*.

b. Saran

Adapun beberapa saran yang dapat diajukan untuk penelitian selanjutnya adalah sebagai berikut:

1. Pengujian kinerja *load balancing* *Web server* dilakukan pada *server* fisik maupun *Virtual Private server* (VPS).
2. Implementasi menggunakan *database server* yang terpusat sehingga tidak terjadi perbedaan data pada setiap *server*.
3. Menggunakan beberapa *Tools* ataupun aplikasi pengujian *Web server* untuk sehingga hasil pengujian kinerja dapat lebih akurat.

DAFTAR PUSTAKA

- [1] K. Wibowo, I. Fitri, and D. Hidayatullah, "Implementasi Load Balancing Web Server Menggunakan Apache di Ubuntu 16.04.," *Sisfotenika*, vol. 10, no. 1, p. 50, 2020, doi: 10.30700/jst.v10i1.773.
- [2] A. Rahmatulloh and F. MSN, "Implementasi Load Balancing Web Server menggunakan Haproxy dan Sinkronisasi File pada Sistem Informasi Akademik Universitas Siliwangi," *J. Nas. Teknol. dan Sist. Inf.*, vol. 3, no. 2, pp. 241–248, 2017, doi: 10.25077/teknosi.v3i2.2017.241-248.
- [3] "Load Balancing - KeyCDN Support," Oct. 04, 2018. <https://www.keycdn.com/support/load-balancing> (accessed May 05, 2023).
- [4] D. K. Hakim, A. Triwidyanto, and U. M. Purwokerto, "BEBAN KOMPUTASI WEB SERVER DENGAN METODE VIRTUALISASI ALGORITHM TESTING ON HAPROXY TO SHARE THE COMPUTATION," no. November, pp. 53–68, 2018.
- [5] A. Saputra, "Mengenal Grafana InfluxDB dan Telegraf Sebagai Monitoring Server," May 14, 2018. <https://adisaputra-id.medium.com/mengenal-grafana-influxdb-dan-telegraf-sebagai-monitoring-server-6c88d3756df2> (accessed May 07, 2023).
- [6] M. Rusli, C. M. Usman, M. F. Mulya, and T. W. Widyaningsih, "Aplikasi Sistem Monitoring Server Menggunakan Device Orange Pi Berbasis Web Service Studi Kasus PT. MNC Televisi Indonesia – MNC Group," *J. SISKOM-KB (Sistem Komput. dan Kecerdasan Buatan)*, vol. 5, no. 2, pp. 24–35, 2022, doi: 10.47970/siskom-kb.v5i2.276.
- [7] M. A. Novianto and S. Munir, "Jurnal Informatika Terpadu," *J. Inform. Terpadu*, vol. 8, no. 2, pp. 47–61, 2022, [Online]. Available: <https://journal.nurulfikri.ac.id/index.php/JIT>
- [8] M. Syani and B. Saputro, "Implementasi Remote Monitoring Pada Virtual Private Server Berbasis Telegram Bot Api (Studi Kasus Politeknik Tedc Bandung)," *J. SISKOM-KB (Sistem Komput. dan Kecerdasan Buatan)*, vol. 4, no. 2, pp. 94–111, 2021, doi: 10.47970/siskom-kb.v4i2.190.
- [9] R. Juniyantara Putra, N. Putra Sastra, and D. M. Wiharta, "Pengembangan Komunikasi Multikanal Untuk Monitoring Infrastruktur Jaringan Berbasis Bot Telegram," *J. SPEKTRUM*, vol. 5, no. 2, p. 152,