Implementasi Load Balancer Pada Flask Web App Untuk Meningkatkan Performansi Web Server

1st Helen Silvia
S1 Teknik Telekomunikasi
Universitas Telkom Purwokerto
Purwokerto, Indonesia
helensilvia@student.telkomuniversity.ac.id

2nd Jafar Gusti Amri Ginting, S.T., M.T.
S1 Teknik Telekomunikasi
Universitas Telkom Purwokerto
Purwokerto, Indonesia
jafargustiamri@telkomuniversity.ac.id

3rd Bongga Arifwidodo, S.S.T., M.T.
S1 Teknik Telekomunikasi
Universitas Telkom Purwokerto
Purwokerto, Indonesia
bonggaa@telkomuniversity.ac.id

***Abstrak* — Server merupakan perangkat yang menyediakan layanan, baik dalam bentuk hardware maupun software. Adanya server memungkinkan untuk penyediaan sumber daya seperti data. Fungsi lain dari server adalah menjadi tempat melakukan deploy aplikasi web. Suatu server, sebaiknya diatur agar memiliki availability atau ketersediaan sehingga ketika client membutuhkan data atau ingin mengakses aplikasi web, layanan akan tetap tersedia. Salah satu cara untuk menjaga availability server adalah dengan mengimplementasikan load balancer. Load balancer merupakan layanan yang dapat mendistribusikan traffic ke beberapa server sehingga server tertentu tidak akan kelebihan beban. Load balancer yang layanannya dapat digunakan antara lain adalah NGINX, dengan metode round-robin, least-connection, dan ip-hash. Setiap metode memiliki cara kerja yang berbeda-beda dan dapat dianalisis Quality of Service-nya untuk menentukan metode yang cocok. Selain itu, ketersediaan server dapat dimonitoring secara real-time menggunakan Grafana dan Prometheus untuk memastikan jika server bekerja dengan baik atau ada kendala yang perlu diatas. Untuk itu, dilakukan implementasi load balancer pada flask web app untuk meningkatkan performansi web server, serta mengintegrasikan dengan layanan Grafana dan Prometheus agar dapat dimonitoring. Berdasarkan hasil penelitian, diketahui bahwa load balancer dapat membagi traffic ke 3 VM menggunakan metode round robin, least connection, dan IP hash, serta dimonitoring pada dashboard Grafana dengan data source Prometheus. Dari ketiga metode, throughput terbesar menggunakan metode IP hash (0,571 Mbps Flask C) dan round robin (0,215 Mbps Flask A, 0,281 Flask B, 0,241 Flask C). Lalu packet loss terkecil menggunakan metode least connection (0,5% Flask A, 0,3% Flask B, 0,4% Flask C). Kemudian delay terkecil menggunakan metode round robin (7,362 ms Flask A, 5,549 ms Flask B, 7,992 Flask C). Terakhir jitter terkecil menggunakan metode round robin (4,092 ms Flask A, 1,735 ms Flask B, 3,822 Flask C).**

Kata kunci: load balancer, flask, nginx, grafana, Prometheus.

I. PENDAHULUAN

Website adalah media yang terdiri dari beberapa halaman saling berkaitan dan berfungsi untuk menampilkan informasi dalam berbagai bentuk seperti teks, gambar, video, suara, atau kombinasi semuanya. Website bersifat multiplatform dan dapat diakses dari berbagai perangkat dengan koneksi internet. Saat ini, banyak perusahaan masih memanfaatkan website untuk menampilkan profil perusahaan, menjual produk, atau menyediakan sistem layanan pelanggan [1]. Berdasarkan data Similar Web, terjadi peningkatan signifikan kunjungan pada situs e-commerce Shopee.com, dari 236 juta pengunjung pada Februari 2024 menjadi 275,6 juta pengunjung pada Maret 2024. Peningkatan jumlah pengguna aplikasi web ini menjadikan performansi sebagai faktor kunci yang perlu dipertimbangkan serius [2]. Performansi yang buruk, seperti server down, dapat memberikan dampak negatif secara teknis maupun bisnis. Contohnya, server down terjadi saat pembukaan rekrutmen BUMN yang dilaporkan oleh Pikiran-rakyat.com karena tingginya jumlah pelamar mengakses situs tersebut. Kejadian serupa juga terjadi pada website kpu.go.id selama pemilu, yang dikarenakan banyaknya pengguna yang mengakses situs untuk mencari informasi TPS. Hal ini diakibatkan oleh server yang tidak mampu menangani lonjakan traffic [3][4]. Untuk mengatasi kasus server down, diperlukan teknologi seperti load balancer, yang mendistribusikan beban traffic ke beberapa jalur koneksi secara seimbang. Teknologi ini sering digunakan oleh berbagai perusahaan atau lembaga di Indonesia maupun dunia untuk memastikan performansi optimal [5]. Grafana dan Prometheus adalah platform open-source yang dapat digunakan untuk memonitor dan menganalisis data secara real-time. Integrasi kedua aplikasi ini memungkinkan administrator sistem memantau kinerja load balancer dan server secara visual, memberikan analisis yang lebih baik untuk pengelolaan performansi server [6]. Teknologi load balancer dalam pengembangan website membantu menjaga responsivitas saat lonjakan traffic besar terjadi. Flask, sebagai web framework berbasis Python, memungkinkan pengembangan aplikasi web yang terstruktur, scalable, dan optimal dalam performansi. Dengan menggunakan Flask, implementasi load balancer dapat memberikan manfaat signifikan dalam meningkatkan performansi, keandalan, dan skalabilitas website [7].

II. KAJIAN TEORI

Berbagai penelitian menunjukkan bahwa implementasi load balancing dapat meningkatkan performa web server secara signifikan. Penelitian oleh Fahmi Apriliansyah dkk. (2020) menunjukkan bahwa algoritma Least Connection pada Nginx meningkatkan throughput dan menurunkan response time dalam menangani traffic tinggi [8]. Sampurna Dadi Riskiono dan Dedi Darwis (2020) membuktikan bahwa

arsitektur dengan load balancing di lingkungan cloud menghasilkan response time lebih kecil dan Quality of Service (QoS) lebih baik dibandingkan server tunggal [9]. Zaidal Bustomi dkk. (2019) menegaskan bahwa Nginx load balancer mampu mengelola trafik secara efektif di lingkungan virtual, meski membutuhkan pengalihan manual saat terjadi gangguan [10]. Cindy Zefira Afiani dan Heru Nurwarsito (2020) mengembangkan algoritma berbasis penggunaan memori dalam Software Defined Network (SDN) yang lebih efisien dibandingkan algoritma berbasis penggunaan CPU, menghasilkan throughput lebih tinggi, response time lebih rendah, dan efisiensi memori yang lebih baik [11].

A. Load Balancer

Load Balancer adalah teknik dalam jaringan komputer dan sistem terdistribusi yang digunakan untuk mengefisienkan trafik masuk ke beberapa server, memastikan pemerataan sumber daya, meningkatkan performa, dan mendukung high availability. Teknologi ini bertindak sebagai penengah antara klien dan server, menerima permintaan dari klien, menentukan server terbaik menggunakan algoritma tertentu, dan mengembalikan respons ke klien [6]. Selain sebagai penyeimbang beban, load balancing juga berfungsi sebagai pengatur trafik, pengalihan jalur trafik, serta pengukur kesehatan server untuk meningkatkan layanan dan mempermudah pengelolaan [9].

Beberapa metode load balancing meliputi:

1. Round Robin: Mendistristribusikan permintaan secara merata ke seluruh server berdasarkan bobot.
2. Least Connections: Mengarahkan permintaan ke server dengan jumlah koneksi aktif paling sedikit.
3. IP Hash: Menentukan server berdasarkan nilai hash dari alamat IP klien, memastikan permintaan dari IP yang sama diteruskan ke server yang sama, kecuali jika server tersebut tidak tersedia [12].

B. Nginx

Nginx atau Engine-X adalah software open-source web server yang pertama kali dikembangkan oleh Igor Sysoev di Rusia pada tahun 2002, dengan versi awal 0.1.0 dirilis pada Oktober 2004 [12]. Nginx memiliki arsitektur asynchronous dan event-driven yang memberikan kemampuan konkurensi tinggi, memungkinkan penanganan ribuan koneksi secara bersamaan. Igor Sysoev juga mengembangkan Nginx menjadi Nginx Plus, yang mampu menangani ratusan hingga ribuan koneksi dan telah digunakan oleh lebih dari 500 situs web [13].

C. HTTPerf

HTTPerf adalah tools untuk mengukur kinerja web server, memungkinkan administrator mendiagnosis dan memperbaiki masalah yang memengaruhi performa server. Tools ini menghasilkan permintaan HTTP GET, memonitor tingkat respons, serta menyajikan ringkasan statistik untuk analisis. HTTPerf unggul karena mampu menghasilkan dan mempertahankan overload server, mendukung protokol HTTP/1.1 dan SSL, serta memiliki extensibility untuk workload baru [13][14].

D. *Grafana*

Grafana adalah aplikasi platform open-source monitoring untuk visualisasi dan analisis data secara real-time dari berbagai sumber seperti Graphite, InfluxDB, OpenTSDB, Prometheus, Elasticsearch, dan CloudWatch. Grafana memungkinkan pemantauan status layanan pada aplikasi atau server melalui panel seperti grafik, singlestat, tabel, dan teks. Dengan dukungan plugin data source yang lengkap, terutama untuk InfluxDB, Grafana menyediakan editor query yang kaya fitur, anotasi, dan templating queries untuk meningkatkan efisiensi analisis data [6][15].

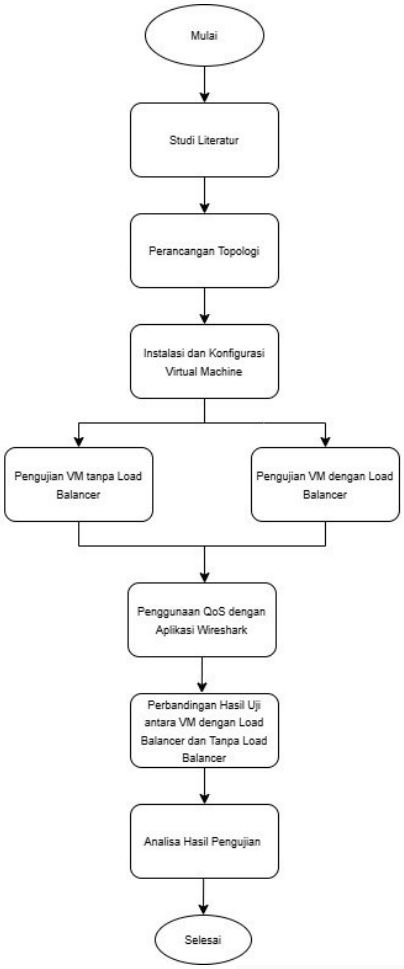
E. *Flask*

Flask adalah web framework berbasis Python yang termasuk jenis microframework, memungkinkan pengembang membuat aplikasi web yang terstruktur dan mudah diatur tanpa memerlukan pustaka tertentu secara default. Meskipun sederhana, Flask dapat diperluas dengan fitur bawaan seperti built-in development server, debugger cepat, dukungan RESTful, templating Jinja2, secure cookies, dan kompatibilitas dengan Google App Engine. Framework ini mendukung unicode, mengikuti WSGI 1.0, serta dilengkapi dokumentasi yang baik dan komunitas diskusi yang luas [16][17].

III. METODE

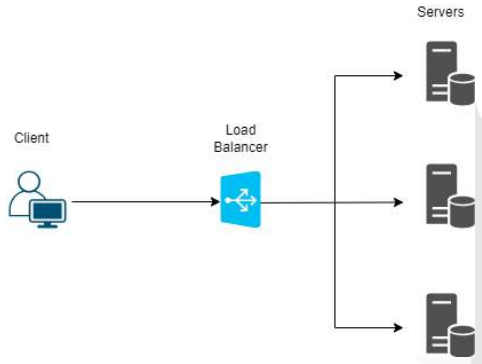
A. Alur Penelitian

Penelitian ini dimulai dengan studi literatur untuk memahami konsep, teknik, dan teknologi terkait load balancing dan performansi web server, diikuti oleh perancangan topologi sistem yang mencakup client, load balancer, dan server. Selanjutnya, dilakukan instalasi dan konfigurasi virtual machine (VM) untuk pengujian, yang dimulai dengan mengukur performansi aplikasi Flask tanpa load balancer sebagai baseline. Pengujian dilanjutkan dengan penerapan load balancer untuk menganalisis dampaknya terhadap performansi aplikasi. QoS diukur menggunakan Wireshark pada kedua skenario, dan hasilnya dibandingkan untuk mengevaluasi efektivitas load balancer. Analisis akhir dilakukan untuk menyimpulkan manfaat implementasi load balancer, dengan mempertimbangkan performansi dan faktor lain yang memengaruhi hasil.



GAMBAR 3. 1 DIAGRAM ALIR PENELITIAN

B. Topologi Jaringan



GAMBAR 3. 2 TOPOLOGI JARINGAN

Pada topologi jaringan (Gambar 3.2), klien mengirimkan permintaan HTTP ke aplikasi web melalui browser atau perangkat lain, yang diterima oleh load balancer sebagai perantara antara klien dan beberapa server backend. Load balancer mendistribusikan lalu lintas jaringan secara merata ke server-server yang menjalankan instance aplikasi Flask, memastikan tidak ada server yang terlalu terbebani dan meningkatkan efisiensi serta performansi aplikasi. Server memproses permintaan dan mengirimkan respons kembali melalui load balancer, dengan implementasi ini juga meningkatkan keandalan sistem melalui pengalihan permintaan ke server lain jika terjadi kegagalan.

IV. PENGUJIAN DAN ANALISIS

A. Pengujian dan Pengumpulan Data

1. Implementasi Load Balancer

Pengujian dilakukan sebanyak 10x dengan mengirimkan masing-masing 1000 request. Setiap pengujian di capture menggunakan Wireshark untuk mendapatkan arah request dari client menuju load balancer, lalu ke server-server. Data pengujian didapatkan dengan memasukkan filter yang berbeda-beda pada masing-masing server seperti terlihat pada Tabel 4.2. Namun, ip.src atau source tetap menggunakan IP dari load balancer dan protocol yang digunakan adalah http.

TABEL 4. 1 FILTER WIRESHARK

Source	Destination	Protocol
192.168.56.103 (*Load balancer*)	192.168.56.109 (*Flask* A)	http
192.168.56.103 (*Load balancer*)	192.168.56.113 (*Flask* B)	http
192.168.56.103 (*Load balancer*)	192.168.56.110 (*Flask* C)	http

2. Monitoring dengan Grafana dan Prometheus

Untuk menguji keakuratan dari dashboard Grafana, dilakukan black-box testing dengan mengirimkan request dan melihat penambahan perubahan pada masing-masing visualisasi. Pengujian dilakukan sebanyak 7x dengan target berbeda-beda sesuai dengan fitur yang ada. Adapun daftar fitur serta target yang diharapkan dapat dilihat pada Tabel 4.3 berikut.

TABEL 4. 2 FITUR PENGUJIAN DASHBOARD GRAFANA

Fitur	Target
Visualisasi State VM A	Dapat menampilkan state UP atau DOWN pada VM
Visualisasi State VM B	Dapat menampilkan state UP atau DOWN pada VM
Visualisasi State VM C	Dapat menampilkan state UP atau DOWN pada VM
Visualisasi Total *Traffic* HTTP pada VM A	Dapat menampilkan total *traffic* HTTP sesuai dengan jumlah *request* yang dikirimkan

Fitur	Target
Visualisasi Total *Traffic* HTTP pada VM B	Dapat menampilkan total *traffic* HTTP sesuai dengan jumlah *request* yang dikirimkan
Visualisasi Total *Traffic* HTTP pada VM C	Dapat menampilkan total *traffic* HTTP sesuai dengan jumlah *request* yang dikirimkan
Visualisasi *Request*/Second seluruh VM	Dapat menampilkan total *request*/second pada seluruh VM secara bersamaan

3. Perbandingan QoS Menggunakan dan Tanpa Load balancer

Untuk memastikan efisiensi penggunaan load balancer, dibutuhkan perbandingan ketika menggunakan dan tidak menggunakan load balancer. Pengujian dilakukan sebanyak 10x dengan mengirimkan masing-masing 1000 request dari HTTPerf ke sebuah server secara langsung dan ke load balancer.

1) Tanpa Load balancer (Flask A)

a. Flask A

Data pengujian yang diperoleh ketika Flask A dijalankan tanpa menggunakan load balancer dapat dilihat pada Tabel 4.4. Rata-rata throughput ketika tidak menggunakan load balancer yaitu sebesar 1,05229 Mbps, rata-rata packet loss sebesar 5,6 packets, rata-rata delay sebesar 1,013826 ms, dan rata-rata Jitter sebesar 0,008496 ms.

TABEL 4. 3 DATA PENGUJIAN TANPA LOAD BALANCER (FLASK A)

Tanpa *Load balancer* (*Flask* A)				
Pengujian ke-	***Throughput* (Mbps)**	***Packet loss* (packets)**	***Delay* (ms)**	***Jitter* (ms)**
1	1,064000	6	1,000549	0,006936
2	1,066132	4	1,001234	0,001468
3	0,927637	4	0,999023	0,000544
4	1,073663	6	1,14795	0,011109
5	1,062937	6	0,991791	0,031553
6	1,065065	7	1,001751	0,00817
7	1,066132	6	1,000199	0,000804
8	1,067202	8	0,999331	0,001343
9	1,066132	4	0,997724	0,001269
10	1,064000	5	0,998704	0,021765
Rata-Rata	1,05229	5,6	1,013826	0,008496

b. Flask B

Data pengujian yang diperoleh ketika Flask B dijalankan tanpa menggunakan load balancer dapat dilihat pada Tabel 4.5. Rata-rata throughput ketika tidak menggunakan load balancer yaitu sebesar 1,034258 Mbps, rata-rata packet loss sebesar 5,9 packets, rata-rata delay sebesar 1,034914 ms, dan rata-rata Jitter sebesar 0,002334 ms.

TABEL 4. 4 DATA PENGUJIAN TANPA LOAD BALANCER (FLASK B)

Tanpa *Load balancer* (*Flask* B)				
Pengujian ke-	***Throughput* (Mbps)**	***Packet loss* (packets)**	***Delay* (ms)**	***Jitter* (ms)**
1	1,078014	6	0,987672	0,000626
2	1,066132	4	0,999066	0,005203
3	1,064000	5	0,998865	0,000123
4	1,065065	6	1,001136	0,001258
5	1,074747	6	0,99965	0,003347
6	1,064000	7	0,99092	0,000558
7	0,942427	6	1,000723	0,000063
8	1,064000	7	1,129776	0,001718
9	0,858065	5	1,000574	0,001252
10	1,066132	7	1,240755	0,009193
Rata-Rata	1,034258	5,9	1,034914	0,002334

c. Flask C

Data pengujian yang diperoleh ketika Flask C dijalankan tanpa menggunakan load balancer dapat dilihat pada Tabel 4.6. Rata-rata throughput ketika tidak menggunakan load balancer yaitu sebesar 1,06261 Mbps, rata-rata packet loss sebesar 5,1 packets, rata-rata delay sebesar 1,002174 ms, dan rata-rata Jitter sebesar 0,002554 ms.

TABEL 4. 5 DATA PENGUJIAN TANPA LOAD BALANCER (FLASK C)

Tanpa *Load balancer* (*Flask* C)				
Pengujian ke-	***Throughput* (Mbps)**	***Packet loss* (packets)**	***Delay* (ms)**	***Jitter* (ms)**
1	1,071501	7	0,993775	0,002312
2	1,068273	4	0,999042	0,001009
3	1,061876	5	0,997357	0,000531

Tanpa *Load balancer* (*Flask* C)				
Pengujian ke-	*Throughput* (Mbps)	*Packet loss* (packets)	*Delay* (ms)	*Jitter* (ms)
4	1,062937	7	1,00294	0,006931
5	1,040078	5	1,001714	0,0018
6	1,063234	4	1,023586	0,005661
7	1,069347	4	1,000645	0,000957
8	1,065065	7	0,996036	0,002679
9	1,057654	4	0,999524	0,000419
10	1,066132	4	1,007122	0,003244
Rata-Rata	1,06261	5,1	1,002174	0,002554

2) Menggunakan Load balancer

a. Flask A

Data pengujian yang diperoleh ketika Flask A dijalankan dengan menggunakan load balancer dapat dilihat pada Tabel 4.7. Rata-rata throughput ketika menggunakan load balancer yaitu sebesar 0,215344 Mbps, rata-rata packet loss sebesar 0,4 packets, rata-rata delay sebesar 7,362247 ms, dan rata-rata Jitter sebesar 4,091509 ms.

TABEL 4. 6 DATA PENGUJIAN DANGAN LOAD BALANCER (FLASK A)

Dengan *Load balancer* (*Flask* A)				
Pengujian ke-	*Throughput* (Mbps)	*Packet loss* (packets)	*Delay* (ms)	*Jitter* (ms)
1	0,35585	0	3,290861	0,003441
2	0,222253	0	5,279653	2,297511
3	0,083156	1	14,14953	11,189
4	0,160841	1	7,35407	1,656378
5	0,08419	0	13,97623	11,01242
6	0,391447	0	2,997264	0,01175
7	0,119069	0	9,883184	6,86139
8	0,157452	2	7,06621	4,082411
9	0,404261	0	2,899381	0,004521
10	0,174925	0	6,72609	3,796271
Rata-Rata	0,215344	0,4	7,362247	4,091509

b. Flask B

Data pengujian yang diperoleh ketika Flask B dijalankan dengan menggunakan load balancer dapat dilihat pada Tabel 4.8. Rata-rata throughput ketika menggunakan load balancer yaitu sebesar 0,280818 Mbps, rata-rata packet loss sebesar 0,5 packets, rata-rata delay sebesar 5,549064 ms, dan rata-rata Jitter sebesar 1,735204 ms.

TABEL 4. 7 DATA PENGUJIAN DANGAN LOAD BALANCER (FLASK B)

Dengan *Load balancer* (*Flask* B)				
Pengujian ke-	*Throughput* (Mbps)	*Packet loss* (packets)	*Delay* (ms)	*Jitter* (ms)
1	0,35585	0	3,291461	0,00519
2	0,362187	1	3,264847	0,305601
3	0,389723	2	3,005256	0,014417
4	0,392839	0	2,996712	0,02582
5	0,126235	0	9,321153	6,324223
6	0,129447	0	9,08974	6,102114
7	0,390112	0	3,001934	0,001277
8	0,168495	1	7,015093	3,10933
9	0,390898	1	2,995768	0,088728
10	0,102398	0	11,50867	1,375336
Rata-Rata	0,280818	0,5	5,549064	1,735204

c. Flask C

Data pengujian yang diperoleh ketika Flask C dijalankan dengan menggunakan load balancer dapat dilihat pada Tabel 4.9. Rata-rata throughput ketika menggunakan load balancer yaitu sebesar 0,240939 Mbps, rata-rata packet loss sebesar 0,5 packets, rata-rata delay sebesar 7,992171 ms, dan rata-rata Jitter sebesar 3,822125 ms.

TABEL 4. 8 DATA PENGUJIAN DANGAN LOAD BALANCER (FLASK C)

Dengan *Load balancer* (*Flask* C)				
Pengujian ke-	*Throughput* (Mbps)	*Packet loss* (packets)	*Delay* (ms)	*Jitter* (ms)
1	0,356919	1	3,281793	0,010735
2	0,099983	1	3,238985	0,277688
3	0,143117	0	11,77026	8,798886
4	0,079327	0	8,221222	5,246685
5	0,056237	0	14,83871	11,91182
6	0,138133	1	21,0182	8,563021
7	0,393393	0	8,556793	3,319354
8	0,388555	0	2,991359	0,005309
9	0,39051	1	3,014229	0,004369
10	0,363216	1	2,99016	0,083381
Rata-Rata	0,240939	0,5	7,992171	3,822125

4. Perbandingan Metode Load balancer

Load balancer pada penelitian berikut menggunakan 3 metode, yaitu round-robin, least-connection, dan ip-hash. Setiap metode akan diuji dan dianalisis QoS-nya untuk kemudian dibandingkan.

1) Round-robin

Pengujian dengan metode round-robin dilakukan sebanyak 10x dengan mengirimkan 1000 request per pengujian. Pengujian ini menggunakan hasil pada pengujian sebelumnya karena sama-sama menggunakan metode round-robin dengan scenario serupa pada Tabel 4.7, Tabel 4.8, dan Tabel 4.9.

2) Least-connection

Pengujian dengan metode least-connection dilakukan sebanyak 10x dengan mengirimkan 1000 request per pengujian. Hasil dari pengujian dirangkum sesuai server tujuan masing-masing.

a. Flask A

Data pengujian yang diperoleh menggunakan load balancer dengan metode least-connection pada Flask A dapat dilihat pada Tabel 4.10. untuk metode least-connection pada Flask A, rata-rata QoS yang didapatkan yaitu throughput sebesar 0,251644 Mbps, packet loss sebesar 0,5%, delay sebesar 11,36755 ms, dan Jitter sebesar 7,82195 ms.

TABEL 4. 9 DATA PENGUJIAN METODE LEAST-CONNECTION (FLASK A)

Load balancer Least-connection (Flask A)				
Pengujian ke-	***Throughput* (Mbps)**	***Packet loss* (%)**	***Delay* (ms)**	***Jitter* (ms)**
1	0,022064	2	53,64483	44,9886
2	0,395197	1	2,962911	0,029952
3	0,11746	0	10,01742	7,023697
4	0,068278	0	17,18032	14,24536
5	0,373014	1	3,131316	0,028179
6	0,286935	1	4,098853	1,176565
7	0,086516	0	13,60037	10,64083
8	0,394168	0	2,984699	0,01221
9	0,388553	0	3,013931	0,035748
10	0,384257	0	3,040866	0,038355
Rata-Rata	0,251644	0,5	11,36755	7,82195

b. Flask B

Data pengujian yang diperoleh menggunakan load balancer dengan metode least-connection pada Flask B dapat dilihat pada Tabel 4.11, untuk metode least-connection pada Flask B, rata-rata QoS yang didapatkan yaitu throughput sebesar 0,190715 Mbps, packet loss sebesar 0,3%, delay sebesar 17,95759 ms, dan Jitter sebesar 9,126842 ms.

Tabel 4. 10 Data pengujian metode least-connection (Flask B)

Load balancer Least-connection (Flask B)				
Pengujian ke-	***Throughput* (Mbps)**	***Packet loss* (%)**	***Delay* (ms)**	***Jitter* (ms)**
1	0,012744	0	93,12033	44,62202
2	0,387767	0	3,020486	0,036104
3	0,122228	0	9,63981	3,161277
4	0,076208	1	15,51185	5,620449
5	0,043771	1	26,87825	24,01724
6	0,000131	1	8,991598	3,145839
7	0,390668	0	3,012725	0,04373
8	0,388944	0	3,011642	0,008027
9	0,087542	0	13,43924	10,56578
10	0,397143	0	2,95	0,047944
Rata-Rata	0,190715	0,3	17,95759	9,126842

c. Flask C

Data pengujian yang diperoleh menggunakan load balancer dengan metode least-connection pada Flask C dapat dilihat pada Tabel 4.12, untuk metode least-connection pada Flask B, rata-rata QoS yang didapatkan yaitu throughput sebesar 0,183868 Mbps, packet loss sebesar 0,4%, delay sebesar 13,75474 ms, dan Jitter sebesar 5,752619 ms.

TABEL 4. 11 DATA PENGUJIAN METODE LEAST-CONNECTION (FLASK C)

Load balancer Least-connection (Flask C)				
Pengujian ke-	***Throughput* (Mbps)**	***Packet loss* (%)**	***Delay* (ms)**	***Jitter* (ms)**
1	0,019523	0	60,80916	8,580775
2	0,118773	0	3,021172	0,017827
3	0,063897	1	9,907	6,963181
4	0,388943	0	18,36057	15,40499
5	0,135279	1	3,012423	0,021106
6	0,201498	0	8,696902	5,727229
7	7,26E-05	1	5,840242	2,83013
8	0,137917	1	16,28469	13,3882
9	0,385003	0	8,572045	4,588274
10	0,387776	0	3,043166	0,004481
Rata-Rata	0,183868	0,4	13,75474	5,752619

3) IP-Hash

Pengujian dengan metode IP-Hash dilakukan sebanyak 10x dengan mengirimkan 1000 request per pengujian. Hasil dari pengujian dirangkum sesuai server tujuan masing-masing.

a. Flask A

Data pengujian yang diperoleh menggunakan load balancer dengan metode IP-hash pada Flask A rata-rata QoS yang didapatkan yaitu throughput sebesar 0 Mbps, packet loss sebesar 0%, delay sebesar 0 ms, dan Jitter sebesar 0 ms.

b. Flask B

Data pengujian yang diperoleh menggunakan load balancer dengan metode IP-hash pada Flask A rata-rata QoS yang didapatkan yaitu throughput sebesar 0 Mbps, packet loss sebesar 0%, delay sebesar 0 ms, dan Jitter sebesar 0 ms.

c. Flask C

Data pengujian yang diperoleh menggunakan load balancer dengan metode IP-hash pada Flask C dapat dilihat pada Tabel 4.15, Berdasarkan Tabel 4.15, untuk metode IP-Hash pada Flask C, rata-rata QoS yang didapatkan yaitu throughput sebesar 0,570875 Mbps, packet loss sebesar 4,4%, delay sebesar 5,726034 ms, dan Jitter sebesar 2,821559 ms.

TABEL 4. 12 DATA PENGUJIAN METODE IP-HASH (FLASK C)

Load balancer IP-Hash (*Flask* C)				
Pengujian ke-	***Throughput*** **(Mbps)**	***Packet loss*** **(%)**	***Delay*** **(ms)**	***Jitter*** **(ms)**
1	0,787946	6	1,485547	0,484633
2	0,213697	4	0,999901	0,003066
3	0,738034	3	5,487704	2,32853
4	0,844441	3	1,584824	0,586547
5	1,171515	2	13,90725	4,12569
6	0,292	7	0,997899	0,000393
7	0,070521	4	4,016119	2,708809
8	0,271334	3	16,65419	12,13579
9	0,150092	5	4,322202	2,984486
10	1,169169	7	7,8047	2,857643
Rata-Rata	0,570875	4,4	5,726034	2,821559

4) Analisis Dan Pembahasan

a. Implementasi Load balancera

Pada implementasi load balancer, seluruh hasil pengujian yang telah dilakukan sebelumnya dirangkum dan disusun secara sistematis dalam Tabel 5.1 di bawah ini. Penyajian ini bertujuan untuk memberikan gambaran yang lebih jelas mengenai hasil pengujian yang telah dilakukan.

TABEL 5. 1 JUMLAH REQUEST

Pengujian Ke-	***Request*** **yang Dikirim**	***Request*** **Diterima** ***Flask*** **A**	***Request*** **Diterima** ***Flask*** **B**	***Request*** **Diterima** ***Flask*** **C**	**Total** ***Request*** **Diterima**
1	1000	333	333	334	1000
2	1000	333	334	333	1000
3	1000	332	334	334	1000
4	1000	333	332	335	1000
5	1000	332	333	335	1000
6	1000	334	333	333	1000
7	1000	332	335	333	1000
8	1000	333	334	333	1000
9	1000	334	333	333	1000
10	1000	334	333	333	1000

Berdasarkan pengujian, total request yang masuk ke ketiga server linear atau selaras dengan jumlah request yang dikirimkan. Setiap mengirimkan 1000 request, maka total request masuk pada ketiga server juga 1000, terlepas dari server tujuan dari request yang dikirimkan.

Berdasarkan Tabel 5.1, hasil analisis yang didapatkan yaitu load balancer mampu menerima request dari client, kemudian diteruskan ke server-server secara merata sesuai jumlah request yang diterima. Traffic tersebut terbagi sama rata karena cara kerja dari metode round-robin yang merupakan metode default load balancer NGINX. Walaupun ada beberapa traffic yang tidak merata seperti pengujian 3, 4, 5, dan 7, hal tersebut dapat dikarenakan server tujuan belum dapat menerima traffic Kembali sehingga perlu diarahkan ke server lainnya.

b. Monitoring dengan Grafana dan Prometheus

Hasil dari proses pengumpulan data yang telah dijelaskan pada sub-bab 4.3 sebelumnya kemudian dirangkum dan ditampilkan dalam Tabel 5.2 di bawah ini untuk memudahkan pemahaman dan analisis selanjutnya.

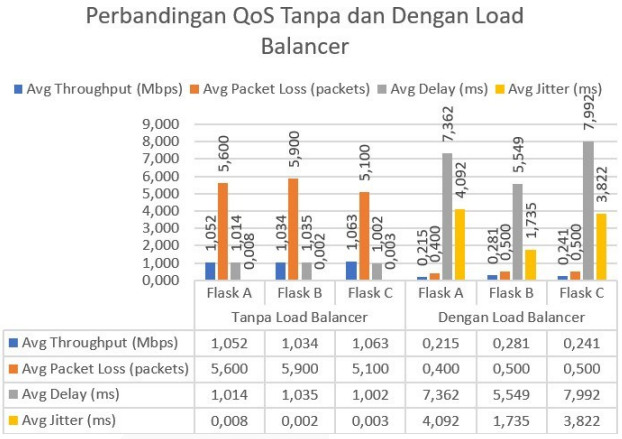
TABEL 5. 2 HASIL PERUBAHAN PADA MASING-MASING VM

Fitur	**Target**	**Hasil**	**Keterangan**
Visualisasi State VM A	Dapat menampilkan state UP atau DOWN pada VM	Visualisasi dapat menampilkan state UP saat VM aktif dan state DOWN saat VM tidak aktif	Tercapai

Fitur	Target	Hasil	Keterangan
Visualisasi State VM B	Dapat menampilkan state UP atau DOWN pada VM	Visualisasi dapat menampilkan state UP saat VM aktif dan state DOWN saat VM tidak aktif	Tercapai
Visualisasi State VM C	Dapat menampilkan state UP atau DOWN pada VM	Visualisasi dapat menampilkan state UP saat VM aktif dan state DOWN saat VM tidak aktif	Tercapai
Visualisasi Total *Traffic* HTTP pada VM A	Dapat menampilkan total *traffic* HTTP sesuai dengan jumlah *request* yang dikirimkan	Visualisasi dapat menampilkan jumlah *traffic* yang sama banyak dengan jumlah *request* yang dikirimkan	Tercapai
Visualisasi Total *Traffic* HTTP pada VM B	Dapat menampilkan total *traffic* HTTP sesuai dengan jumlah *request* yang dikirimkan	Visualisasi dapat menampilkan jumlah *traffic* yang sama banyak dengan jumlah *request* yang dikirimkan	Tercapai
Visualisasi Total *Traffic* HTTP pada VM C	Dapat menampilkan total *traffic* HTTP sesuai dengan jumlah *request* yang dikirimkan	Visualisasi dapat menampilkan jumlah *traffic* yang sama banyak dengan jumlah *request* yang dikirimkan	Tercapai
Visualisasi *Request*/Second seluruh VM	Dapat menampilkan total *request*/second pada seluruh VM secara bersamaan	Visualisasi dapat menampilkan total *request*/second pada seluruh VM secara bersamaan	Tercapai

c. Perbandingan QoS Menggunakan dan Tanpa Load balancer

Pada pengujian perbandingan QoS menggunakan dan tanpa menggunakan load balancer, setiap rata-rata hasil pengujian dikumpulkan dan dibandingkan. Selanjutnya, data tersebut divisualisasikan untuk mempermudah analisis hasil. Secara keseluruhan, perbandingan QoS pada saat tidak menggunakan load balancer dan saat menggunakan load balancer dapat dilihat pada Gambar 5.4 berikut.



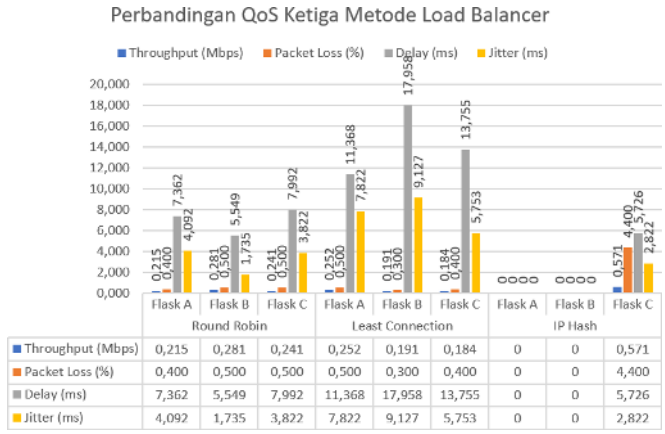
	Tanpa Load Balancer			Dengan Load Balancer		
	Flask A	Flask B	Flask C	Flask A	Flask B	Flask C
Avg Throughput (Mbps)	1,052	1,034	1,063	0,215	0,281	0,241
Avg Packet Loss (packets)	5,600	5,900	5,100	0,400	0,500	0,500
Avg Delay (ms)	1,014	1,035	1,002	7,362	5,549	7,992
Avg Jitter (ms)	0,008	0,002	0,003	4,092	1,735	3,822

GAMBAR 5. 5 GRAFIK PERBANDINGAN QOS TANPA DAN DENGAN LOAD BALANCER

Berdasarkan analisis grafik, throughput, delay, dan Jitter saat tidak menggunakan load balancer dinilai lebih baik. Namun, ketiga metrik tersebut tidak menunjukkan bahwa mengecualikan load balancer lebih baik karena cara kerja yang dimiliki oleh load balancer secara tidak langsung mempengaruhi hasil pengujian. Sedangkan, nilai packet loss yang dihasilkan lebih baik saat menggunakan load balancer karena traffic yang masuk telah dibagi ke beberapa server. Sehingga, penggunaan load balancer lebih baik daripada tidak menggunakan load balancer.

d. Perbandingan Metode Load balancer

Pada pengujian perbandingan QoS menggunakan load balancer dengan metode round-robin, least-connection, dan IP-hash, setiap rata-rata hasil pengujian dikumpulkan dan dibandingkan. Selanjutnya, data tersebut divisualisasikan untuk mempermudah analisis hasil. Secara keseluruhan, perbandingan QoS pada saat menggunakan load balancer dengan metode round-robin, least-connection, dan ip-hash dapat dilihat pada Gambar 5.5 berikut.



	Round Robin			Least Connection			IP Hash		
	Flask A	Flask B	Flask C	Flask A	Flask B	Flask C	Flask A	Flask B	Flask C
Throughput (Mbps)	0,215	0,281	0,241	0,252	0,191	0,184	0	0	0,571
Packet Loss (%)	0,400	0,500	0,500	0,500	0,300	0,400	0	0	4,400
Delay (ms)	7,362	5,549	7,992	11,368	17,958	13,755	0	0	5,726
Jitter (ms)	4,092	1,735	3,822	7,822	9,127	5,753	0	0	2,822

GAMBAR 5. 10 GRAFIK PERBANDINGAN QOS PADA LOAD BALANCER DENGAN METODE

Berdasarkan analisis grafik, metode IP-Hash menunjukkan throughput tertinggi, packet loss tertinggi, delay terendah, dan jitter terendah, karena mekanismenya yang memfokuskan traffic ke server yang sama secara konsisten untuk setiap client. Sementara itu, metode round-robin cenderung lebih stabil dalam throughput, tetapi memiliki packet loss lebih tinggi dibandingkan least-connection. Sebaliknya, least-connection menghasilkan delay dan jitter yang lebih tinggi akibat fokusnya pada koneksi aktif, yang menyebabkan pembagian traffic kurang merata. Hal ini menunjukkan bahwa setiap metode memiliki keunggulan dan kekurangan yang bergantung pada distribusi traffic yang digunakan.

V. KESIMPULAN

Berdasarkan hasil pengujian dan analisis, maka dapat diambil beberapa kesimpulan sebagai berikut.

1. Load balancer dapat diimplementasikan menggunakan NGINX pada VM Ubuntu Server. Request yang dikirimkan ke load balancer telah berhasil diarahkan ke server-server upstream pada konfigurasi.
2. Monitoring dapat dilakukan pada dashboard Grafana dan Prometheus. Seluruh fitur yang telah didesain pada dashboard seperti visualisasi state per VM, visualisasi total request http per VM, dan total request/second seluruh VM, dapat digunakan sebagai visualisasi dalam melakukan monitoring load balancer.
3. Pada pengujian perbandingan QoS saat tidak menggunakan load balancer dan saat menggunakan load balancer, nilai throughput, delay, dan jitter saat tidak menggunakan load balancer dinilai lebih baik. Namun, hasil tersebut berasal dari cara kerja load balancer, bukan kekurangannya. Kemudian, nilai packet loss yang dihasilkan lebih baik saat menggunakan load balancer karena traffic yang masuk telah dibagi ke beberapa server. Sehingga, penggunaan load balancer lebih baik daripada tidak menggunakan load balancer.
4. Pada pengujian perbandingan QoS untuk metode round robin, least connection, dan ip hash, hasil menunjukkan IP Hash dengan throughput tertinggi karena traffic hanya difokuskan ke satu server, dan round robin karena traffic dibagi sama rata dan cederung stabil. Lalu, packet loss terkecil menggunakan metode least connection karena memfokuskan pada koneksi yang aktif. Kemudian, delay terkecil menggunakan metode round robin karena traffic dibagi sama rata yang membuat jarak antara traffic semakin kecil. Terakhir, jitter terkecil menggunakan metode round robin karena kecilnya ragam variansi delay dibandingkan metode lainnya. Sehingga, penggunaan metode round robin secara keseluruhan lebih baik dibandingkan metode least connection dan IP hash.

VI. REFERENSI

[1] S. Sonny and S. N. Rizki, “Pengembangan Sistem Presensi Karyawan dengan Teknologi GPS Berbasis WEB,” J. Comasie, vol. 6, no. 2, p. 3, 2021.

[2] SimilarWeb, “Global Website Traffic Rankings,” SimilarWeb, 2024. https://www.similarweb.com/ (accessed Jun. 05, 2024).

[3] B. Sugara, “Trafik Tinggi di Hari Pertama, Calon Pelamar Kesulitan Login di Situs Web Rekrutmen Bersama BUMN 2024,” Pikiran Rakyat Tengerang Kota, 2024. https://tangerangkota.pikiran-rakyat.com/metropolitan/pr-3477879088/trafik-tinggi-di-hari-pertama-calon-pelamar-kesulitan-login-di-situs-web-rekrutmen-bersama-bumn-2024?page=all (accessed Jun. 05, 2024).

[4] T. R. Depok, “Website KPU Down di Hari Pemilu 2024, Tak Kuat Menerima Traffic yang Begitu Besar,” Rublik Depok Com, 2024. https://depokraya.pikiran-rakyat.com/politik/pr-3297717585/website-kpu-down-di-hari-pemilu-2024-tak-kuat-menerima-traffic-yang-begitu-besar?page=all (accessed Jun. 05, 2024).

[5] B. D. P. Joko Iskandar, “Analisis Teknik Load Balancing Metode Per Connection Classifier (PCC) untuk Pembagian Beban Kerja Server,” G-Tech J. Teknol. Terap., vol. 8, no. 1, pp. 186–195, 2024.

[6] R. T. Darmawan, “Implementasi Grafana untuk Auto-Scaling pada NGINX Web-Server di Lingkungan Docker Container,” Pengemb. Teknol. Inf. dan Ilmu Komput., vol. 7, no. 7, pp. 3163–3167, 2023.

[7] F. Documentation, “Flask,” Flask Documentation. https://flask.palletsprojects.com/en/3.0.x/ (accessed Apr. 28, 2020).

[8] F. Apriliansyah, I. Fitri, and A. Iskandar, “Implementasi Load Balancing Pada Web Server Menggunakan Nginx,” J. Teknol. dan Manaj. Inform., vol. 6, no. 1, 2020, doi: 10.3997/2214-4609.201801770.

[9] S. D. Riskiono and D. Darwis, “Peran Load Balancing Dalam Meningkatkan Kinerja Web Server Di Lingkungan Cloud,” Krea-TIF, vol. 8, no. 2, p. 1, 2020, doi: 10.32832/kreatif.v8i2.3503.

[10] Z. Bustomi, M. Syahiruddin, M. I. Afandi, and K. F. Holle, “Load Balancing Web Server Menggunakan Nginx pada Lingkungan Virtual,” J. Inform. J. Pengemb. IT, vol. 5, no. 1, pp. 32–36, 2020, doi: 10.30591/jpit.v5i1.1745.

[11] Cindy Zefira Afiani1, and Heru Nurwarsito, “Implementasi Load Balancing Web Server Pada POX Controller Berbasis Penggunaan Memori Dengan Agen Psutil Pada Software Defined Network,” vol. 5, No. 1 115-123, 2021

[12] N. Docs, “HTTP Load Balancing,” NGINX Docs. https://docs.Nginx.com/Nginx/admin-guide/load-balancer/http-load-balancer/ (accessed Jun. 05, 2024).

[13] R. B. Kisnandar, "Analisis Perbandingan Kinerja Web Server NGINX, Apache, dan LIGHTTPD Dengan Metode Stress Test," J. Karya Ilm. Civ. Akad. UTDI, vol. 2, no. 2017, pp. 7–15, 2019.
[14] D. Puri, "Mengenal Apa itu Performance Testing Beserta dengan Toolsnya," Binus University, 2021. https://sis.binus.ac.id/2021/09/15/mengenal-apa-itu-performance-testing-beserta-dengan-toolsnya/#:~:text=httperf adalah salah satu tools,Workloads untuk mengukur kinerja server (accessed Jun. 05, 2024).
[15] D. Rahman, H. Amnur, and I. Rahmayuni, "Monitoring Server dengan Prometheus dan Grafana serta Notifikasi Telegram," JITSI J. Ilm. Teknol. Sist. Inf., vol. 1, no. 4, pp. 133–138, 2020, doi: 10.30630/jitsi.1.4.19.
[16] R. Irsyad, "Penggunaan Python Web Framework Flask Untuk Pemula," Khazanah Inform. J. Ilmu Komput. dan Inform., vol. 4, no. 1, pp. 8–15, 2018.
[17] Dqlab.id, "Mengenal Flask, Library Machine Learning Python Idaman Developer," dqlab.id, 2021. https://dqlab.id/mengenal-flask-library-machine-learning-python-idaman-developer (accessed Jun. 05, 2024).