Analisis Performansi Algoritma Load Balancing Least Connection dan Weighted Roud Robin Berbasis Goole Cloud Platform

1st Oktavia Ayu Andini
Direktorat Kampus Purwokerto
Universitas Telkom Purwokerto
Purwokerto, Indonesia
oktaviaayuandini@telkomuniversity.ac.id

2nd Iqsyahiro Kresna
Direktorat Kampus Purwokerto
Universotas Telkom Purwokerto
Purwokerto, Indonesia
hiroka @telkomuniversity.ac.id

***Abstrak* — Pertumbuhan penggunaan layanan berbasis internet yang meningkat dalam beberapa tahun terakhir menuntut sistem server memiliki kemampuan distribusi beban yang optimal. Permasalahan utama yang muncul adalah penurunan performa akibat *overload* pada server. Untuk mengatasi hal ini, penelitian ini melakukan perbandingan terhadap dua algoritma *load balancing*, yaitu *Least Connection* dan *Weighted Round Robin* (WRR), yang diimplementasikan menggunakan HAProxy di atas infrastruktur Google Cloud Platform. Pengujian dilakukan dengan menggunakan apache benchmark, dengan variasi jumlah request 1000 hingga 5000 dan jumlah koneksi bersamaan 20 hingga 100. Hasil pengujian menunjukkan bahwa algoritma *Weighted Round Robin* mampu menurunkan *delay* hingga 0,317 ms, sedangkan algoritma *Least Connection* mampu mempertahankan throughput lebih tinggi sebesar 3181,824 KB/s pada beban maksimum. Selain itu, jitter pada algoritma Weighted Round Robin tercatat lebih rendah, mencapai hanya 0,001 ms. Penelitian ini memberikan kontribusi berupa analisis performansi berbasis Quality of Service (QoS), yang dapat menjadi acuan dalam pemilihan metode *load balancing* untuk sistem berbasis google cloud platform.**

***Kata kunci— Load balancing, least connection, HAProxy, weighted round robin,* Google Cloud Platform**

I. PENDAHULUAN

Besarnya jumlah pengguna internet di Indonesia tentu berpengaruh terhadap volume lalu lintas data pada server[1]. Permasalahan yang kerap terjadi adalah kurang optimalnya manajemen server web oleh pengelolanya. Akibatnya, sering terjadi server downtime dan kelebihan beban (overload) ketika server menerima akses secara bersamaan dari banyak pengguna, terutama jika hanya menggunakan satu server untuk menangani seluruh permintaan. Oleh karena itu, dibutuhkan sebuah sistem server yang dapat menangani banyak akses sekaligus dan dapat di tingkatkan skalabilitasnya sesuai dengan kebutuhan. Sistem yang dapat memenuhi kebutuhan tersebut adalah server *load balancing*.

Load balancing merupakan proses membagi beban lalu lintas jaringan yang masuk ke beberapa server, instant, atau sumber daya untuk memastikan tidak ada satu pun sumber daya yang kelebihan beban (overload) atau kewalahan [2]. Dengan menggunakan load balancing, situs web dapat tetap online dan stabil walaupun mengalami peningkatan lalu lintas data karena banyaknya pengunjung. Tujuan utama dari load balancing adalah untuk mengurangi beban pada server tertentu, meningkatkan ketersediaan layanan, dan memastikan sumber daya server digunakan secara optimal. Dengan pembagian beban secara merata, load balancing dapat mencegah server menjadi terlalu sibuk, sehingga kinerja keseluruhan sistem dapat dipertahankan. Jika salah satu server down atau tidak tersedia, load balancer dapat secara otomatis memindahkan lalu lintas ke server lain. Dengan load balancing, sistem dapat dengan mudah menambah atau mengurangi server sesuai kebutuhan, yang memungkinkan skalabilitas yang lebih baik.

Salah satu perangkat lunak load balancer yang sering digunakan dan open-source adalah HAProxy (High Availability Proxy). HAProxy mampu membagi permintaan dari pengguna ke server web menggunakan berbagai strategi algoritma, dua diantaranya yaitu Least Connection dan Weighted Round Robin (WRR). Algoritma Least Connection akan mengarahkan permintaan dari pengguna ke server dengan jumlah koneksi aktif terendah, dengan harapan mengoptimalkan penggunaan sumber daya yang ada, sedangkan WRR mendistribusikan permintaan berdasarkan bobot (weight) yang ditetapkan pada masing-masing server. Kedua algoritma ini memiliki karakteristik berbeda dan dapat mempengaruhi performa layanan secara signifikan. Namun, implementasi algoritma ini memerlukan analisis lebih lanjut untuk memahami bagaimana performansi sistem dapat dipengaruhi oleh penggunaan algoritma ini.

Dalam pengaplikasiannya, load balancing dapat dilakukan pada beberapa platform, salah satunya yaitu pada layanan cloud computing [3]. Cloud computing merupakan suatu model layanan teknologi informasi yang memberikan kemudahan bagi pengguna dalam mengakses dan memanfaatkan sumber daya seperti penyimpanan data, server, basis data, jaringan, dan perangkat lunak dengan sistem pembayaran berdasarkan penggunaan [4]. Cloud computing memungkinkan pengguna untuk mengatur sumber daya IT mereka dengan lebih efisien, elastis, dan hemat biaya. Beberapa penyedia cloud load balancing yang terkenal antara lain AWS, Google Cloud Platform, Alibaba Cloud dan

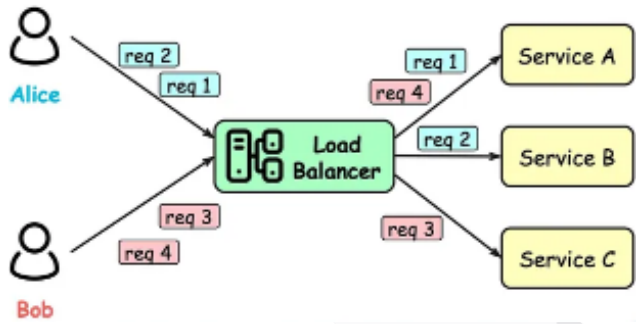
Microsoft Azure. Masing-masing memiliki fitur dan opsi yang berbeda untuk menunjang kebutuhan load balancing yang berbeda.

II. KAJIAN TEORI

Penelitian ini telah melakukan studi literatur terhadap jurnal yang selaras dengan tema penelitian. Penelitian ini menggunakan lima jurnal yang digunakan sebagai kajian literatur untuk digunakan sebagai bahan acuan dalam melakukan penelitian ini, diantaranya yaitu terdapat tiga jurnal nasional dan dua jurnal internasional dengan rentang tahun antara tahun 2020 hingga tahun 2023.

A. Load Balancing

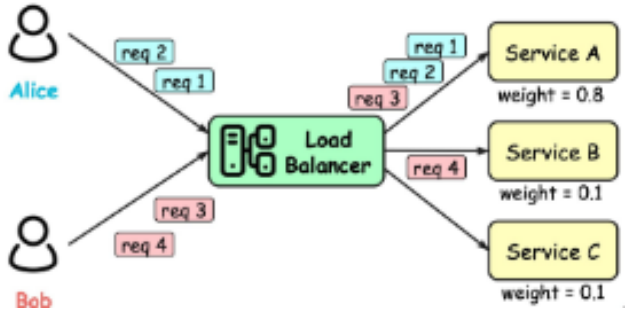
Load balancing adalah suatu teknik dalam jaringan komputer yang mempunyai sistem kerja dengan cara membagi permintaan yang masuk ke sejumlah komputer atau cluster dengan tujuan untuk mencapai pemanfaatan yang optimal dari sumber daya yang ada, memperkecil waktu respon, memperbesar throughput , serta mengurangi terjadinya overload[4]. Jaringan komputer menggunakan teknik load balancing untuk membagi beban kerja di antara berbagai server atau sumber daya lainnya untuk meningkatkan kapasitas dan kehandalan sistem. Sebuah load balancer memerlukan metode pengaturan dalam proses perhitungannya untuk melakukan proses load balancing. Beberapa algoritma load balancing yang umum digunakan adalah Round Robin, Weighted Round Robin, Least Connections, dan Least Response Time.



GAMBAR 1
ROUND ROBIN [5]

GAMBAR 1

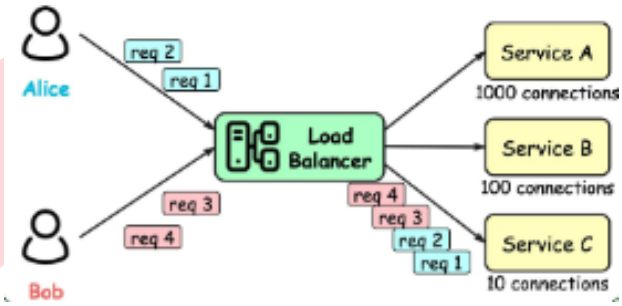
ROUND *ROBIN* menunjukkan prinsip kerja algoritma Round Robin adalah dengan mendistribusikan permintaan ke server-server yang tersedia secara bergantian [6]. Tidak peduli jumlah koneksi atau beban server, load balancer akan meneruskan permintaan ke server berikutnya secara berurutan dalam daftar. Saat semua server sudah menerima satu permintaan, proses akan diulang dari awal.



GAMBAR 2
WEIGHTED ROUND ROBIN [5]

GAMBAR 2

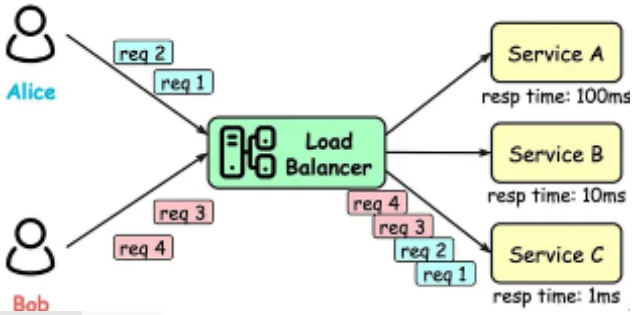
WEIGHTED ROUND *ROBIN* menunjukkan bahwa algoritma weighted round robin memiliki konsep hampir sama dengan algoritma round robin, akan tetapi diberikan tambahan kondisi baru yaitu mempertimbangkan beban server dan memberikan jumlah tugas yang lebih tinggi ke server yang mempunyai kapasitas yang lebih tinggi [7]. Permintaan akan dikirim ke server secara bergilir (round-robin) oleh load balancer, tetapi jumlah permintaan yang dikirim ke masing-masing server akan proporsional dengan bobotnya. Server dengan bobot yang lebih besar akan menerima lebih banyak permintaan daripada server dengan bobot yang lebih kecil. Sampai semua server mendapatkan giliran, pengiriman permintaan akan berulang.



GAMBAR 3
LEAST CONNECTION [5]

GAMBAR 3

LEAST *CONNECTION* menunjukkan bahwa algoritma Least Connection mengirimkan permintaan ke server dengan jumlah koneksi aktif paling sedikit dan memastikan beban kerja didistribusikan secara merata di antara server, serta mencegah overload pada server tertentu [6]. Algoritma Least Connection cocok untuk diterapkan pada lingkungan cloud computing seperti Google Cloud, Microsoft Azure, dan Amazon Web Services, di mana skalabilitas dan ketersediaan layanan merupakan faktor penting.



GAMBAR 4
LEAST RESPONS TIME [5]

GAMBAR 4

LEAST RESPONS *TIME* adalah salah satu algoritma load balancing yang mempertimbangkan waktu respons (response time) dari masing-masing server saat mendistribusikan beban kerja [6]. Sebagai contoh, algoritma Least Response Time bekerja dengan cara berikut: Load balancer akan melacak dan mengukur waktu respons (response time) dari setiap server yang tersedia. Saat ada permintaan masuk, permintaan tersebut akan diarahkan ke server dengan waktu respons tercepat, dengan tujuan untuk memilih server dengan waktu respons yang paling cepat untuk memberikan layanan yang lebih baik kepada pengguna.

B. Cloud Computing

Cloud Computing atau komputasi awan adalah gabungan pemanfaatan teknologi komputer (‘komputasi’) dan pengembangan berbasis Internet [8]. Cloud Storage yang mana merupakan metafora dari internet, sebagaimana media penyimpanan yang sering digambarkan pada diagram jaringan komputer, Cloud Storage dalam Cloud Computing juga merupakan abstraksi dari infrastruktur kompleks yang disembunyikannya kapabilitas yang terkait teknologi informasi disajikan sebagai suatu layanan [9]. Cloud computing memungkinkan Anda mengakses sumber daya komputasi ini dari pusat data penyedia cloud, alih-alih memiliki komputer atau server fisik di tempat untuk menangani tugas-tugas ini. Beberapa penyedia cloud populer adalah Amazon Web Services (AWS), Microsoft Azure, dan Google Cloud Platform.

C. Google Cloud Platform

Gambar 2. 5 Google Cloud Platform (GCP) adalah sebuah suite layanan komputasi awan yang disediakan oleh Google. Google Cloud Platform (GCP) menawarkan berbagai layanan yang berkaitan dengan load balancing, salah satunya adalah Google Cloud Load Balancing, yang dapat secara otomatis mendistribusikan lalu lintas web, aplikasi, atau API ke seluruh sumber daya komputasi yang tersedia [9]. GCP juga menyediakan layanan load balancing jaringan, yang menyediakan load balancing pada lapisan jaringan (layer 4) untuk lalu lintas TCP atau UDP dan memiliki berbagai opsi load balancing, seperti global, regional, atau internal. Layanan ini juga dapat melakukan load balancing pada berbagai jenis lalu lintas, termasuk HTTP/HTTPS, TCP, dan UDP, dan dapat mendistribusikan lalu lintas secara cepat dan efisien berdasarkan karakteristik paket jaringan.

D. Web Server

Web server adalah sebuah perangkat lunak yang berfungsi untuk menerima, memproses, dan merespons permintaan dari klien (seperti browser web) melalui protokol HTTP atau HTTPS [10]. Web server dapat digunakan pada berbagai jenis perangkat keras, mulai dari komputer desktop hingga server kelas enterprise. Apache HTTP Server, Nginx, Microsoft Internet Information Services (IIS), dan Google Web Server adalah beberapa contoh web server yang sangat popular. Dalam komputasi awan, web server dapat dijalankan pada infrastruktur yang disediakan oleh platform awan, seperti Google Cloud Platform. Pengguna dapat dengan mudah mengkonfigurasi dan menjalankan web server sesuai kebutuhan, memanfaatkan sumber daya komputasi, jaringan, dan penyimpanan yang disediakan oleh platform awan, memungkinkan perusahaan untuk berkonsentrasi pada pengembangan aplikasi web tanpa harus mengelola infrastruktur web server secara langsung[11].

E. HAProxy

HAProxy adalah penyeimbang beban sumber terbuka populer yang mendistribusikan lalu lintas yang masuk ke beberapa server untuk meningkatkan kinerja aplikasi, skalabilitas, dan keandalan [12]. HAProxy sangat populer digunakan di lingkungan web hosting skala besar karena kemampuannya dalam menangani ribuan hingga jutaan koneksi secara bersamaan. Selain itu, HAProxy menyediakan berbagai algoritma load balancing seperti round robin, least connection, dan weighted round robin, sehingga pengguna dapat memilih metode distribusi beban sesuai kebutuhan.

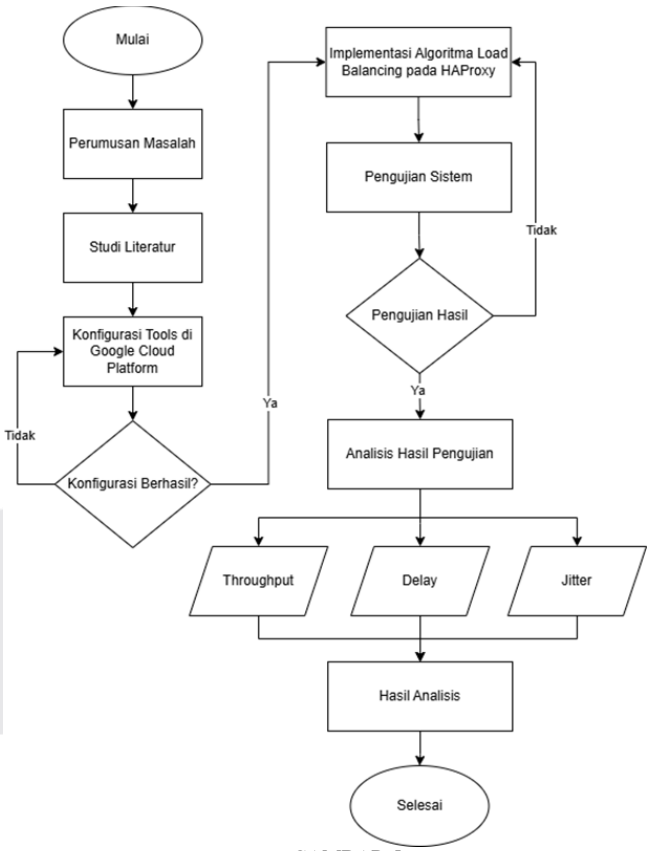
III. METODE

Memberikan Penelitian ini dilakukan dengan penelitian eksperimen, metode penelitian yang bertujuan untuk menjelaskan hubungan sebab-akibat (kausalitas) antara satu variabel dengan lainnya (variabel X dan variabel Y) [13]. Pada penelitian ini dilakukan dengan cara menerapkan, menguji, dan menganalisis dari dua algoritma load balancing, yakni least connection dan weighted round robin (WRR). Tahap akhir dari penelitian ini adalah melakukan analisis terhadap hasil evaluasi dan menarik kesimpulan berdasarkan performa model, serta memberikan saran untuk pengembangan sistem segmentasi lebih lanjut pada masa mendatang.

Bahan yang digunakan dalam penelitian ini berupa data-data yang berasal dari studi literatur dan beberapa skenario pengujian berupa delay, jitter, dan throughput.

A. Diagram Alir Penelitian

Definisikan Penelitian ini melibatkan beberapa tahap penelitian, berikut diagram alir penelitian yang dilakukan:



GAMBAR 5
DIAGRAM ALIR PENELITIAN

Berdasarkan GAMBAR 5 Diagram Alir Penelitian langkah pertama dalam proses penelitian adalah perumusan masalah, di mana penulis mengidentifikasi isu utama yang akan menjadi fokus penelitian. Tahap ini mencakup penetapan pertanyaan penelitian yang relevan, menentukan batasan yang sesuai, merumuskan tujuan penelitian secara jelas, serta

memilih metode yang tepat untuk diterapkan dalam penelitian

Pada tahap studi literatur, peneliti melakukan pencarian dan analisis berbagai sumber referensi yang relevan dengan topik penelitian, khususnya terkait load balancing. Sumber referensi yang dikaji meliputi jurnal ilmiah, buku, artikel, dan literatur terkait lainnya yang menjadi dasar penelitian. Tujuan utamanya adalah untuk mengidentifikasi kontribusi penelitian ini terhadap pengembangan ilmu pengetahuan dan sosial, baik dalam hal teori maupun metode penelitian yang sudah digunakan sebelumnya.

Pada tahap konfigurasi alat dan layanan yang digunakan dalam penelitian pada platform Google Cloud Platform (GCP), tools yang dimaksud berupa Compute Engine (untuk pembuatan dan pengelolaan virtual machine). Tujuan utamanya adalah untuk menyediakan infrastruktur cloud yang stabil dan dapat diakses secara publik untuk keperluan pengujian sistem load balancing. Dengan memanfaatkan fleksibilitas dan skalabilitas dari GCP, penelitian dapat melakukan pengujian secara efisien, terstruktur, dan berulang.

Pada tahap konfigurasi dan penerapan algoritma pada software load balancer HAProxy, yaitu sebuah perangkat lunak open-source yang berfungsi sebagai reverse proxy dan load balancer untuk protokol HTTP dan TCP [14]. Yang mana proses implementasi mencakup penyesuaian file konfigurasi haproxy.cfg. Penyesuaian konfigurasi mencakup penentuan backend server, pendefinisian frontend service, serta pemilihan algoritma load balancer yang akan digunakan.

Dalam file konfigurasi tersebut, peneliti mendeklarasikan masing-masing alamat IP internal dari tiga web server sebagai backend, dan menentukan metode load balancing untuk mendistribusikan permintaan dari pengguna yang diterima melalui port 80. HAProxy memberikan fleksibilitas tinggi dalam hal konfigurasi, sehingga peneliti dapat mengubah metode load balancing secara manual sesuai kebutuhan skenario pengujian. Setelah konfigurasi disimpan, layanan HAProxy dijalankan ulang agar perubahan dapat diterapkan dan sistem siap menerima trafik uji beban.

Pada tahap pengujian sistem akan memastikan bahwa sistem yang telah dikonfigurasi berjalan dengan baik, dengan melakukan verifikasi bahwa server HAProxy mampu membagi beban ke server backend dengan benar. Tujuan utamanya yaitu untuk menilai apakah konfigurasi dan sistem HAProxy sudah dapat menjalankan algoritma load balancing sesuai rancangan, serta memastikan server backend dapat merespons permintaan dari klien tanpa error sebelum pengujian performa dilakukan.

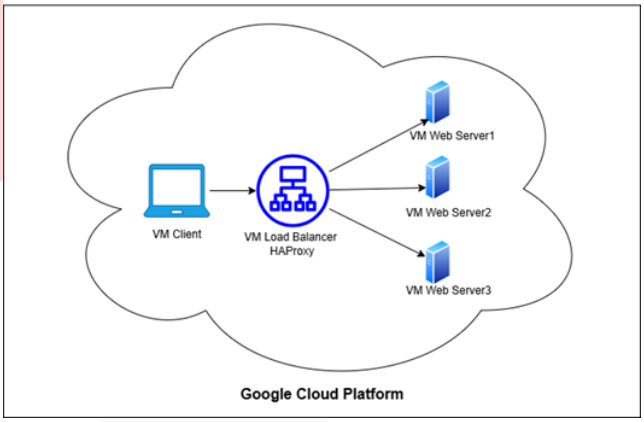
Pengujian hasil dilakukan menggunakan alat bantu seperti Apache Benchmark(ab) dengan memberikan jumlah request dan tingkat konkuren tertentu secara berulang-ulang, baik untuk algoritma least connection maupun weighted round robin. Tujuannya adalah untuk mengumpulkan data kuantitatif terkait performansi sistem, seperti throughput, jitter, dan delay, guna menjadi dasar dalam melakukan analisis perbandingan antar algoritma yang diuji.

IV. HASIL DAN PEMBAHASAN

Sistem terdiri dari lima buah virtual machine (VM) yang dibuat melalui layanan Google Cloud Platform (GCP), yaitu tiga VM sebagai web server, satu VM sebagai load balancer, dan satu VM sebagai pengguna. Pengujian dilakukan menggunakan apache benchmark dengan variasi jumlah request dan concurrent user. Tujuannya adalah untuk mengevaluasi performa algoritma Least Connection dan Weighted Round Robin (WRR) berdasarkan parameter Quality of Service (QoS): throughput, delay, dan jitter. Ketiga parameter ini dipilih karena merepresentasikan efisien, kecepatan, dan kestabilan.

A. Skenario Percobaan

Google Cloud Platform menyediakan berbagai macam fitur seperti Compute Engine, Cloud Storage, VPC, serta sistem monitoring dan keamanan. Dalam penelitian ini, Google Cloud Platform dimanfaatkan sebagai media utama dalam membangun infrastruktur virtual yang digunakan untuk menguji algoritma load balancing.

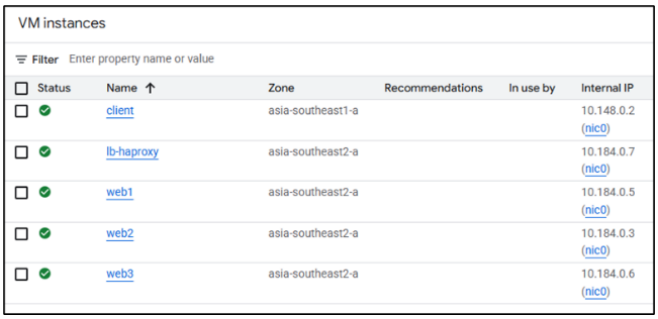


GAMBAR 6
SKEMA PENGUJIAN

Berdasarkan Gambar 6 SKEMA PENGUJIAN menunjukkan bagaimana penelitian ini akan dilaksanakan. Dalam pengujian ini, pengguna mengirimkan permintaan layanan. Permintaan tersebut tidak langsung diteruskan ke server, tetapi terlebih dahulu diterima oleh komponen *Load Balancer*. *Load Balancer* disini berfungsi untuk mendistribusikan permintaan secara merata ke beberapa web server, dengan tujuan mencegah overloading pada satu server saja. *Load balancer* juga bertindak sebagai perantara yang cerdas, yang dapat memilih server berdasarkan algoritma tertentu baik *least connection* maupun *weighted round robin*.

1. Pembuatan Virtual Machine (VM) instance

Pada google cloud platform dibuat lima virtual machine (VM) melalui layanan compute engine, yang terdiri dari 3 VM sebagai web server(web1, web2, web3), 1 VM sebagai load balancer HAProxy, dan 1 VM pengguna untuk memberikan request jumlah beban server.



GAMBAR 7
VIRTUAL MACHINE

Gambar 7

VIRTUAL MACHINE menunjukkan daftar kelima VM yang telah dibuat pada konsol Compute Engine di Google Cloud Platform. Adapun spesifikasi dari masing-masing VM dapat dilihat pada TABEL 1 *SPESIFIKASI VM* berikut :

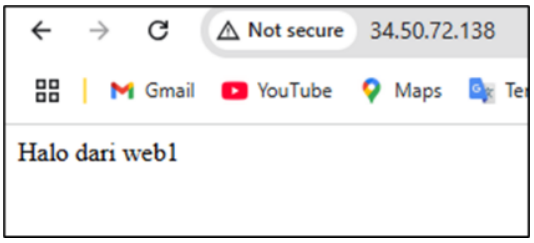
TABEL 1
SPESIFIKASI VM

Nama VM	Jenis Mesin	Sistem Operasi	Software
Pengguna	e2-medium (2 vCPU, 4 GB RAM)	Ubuntu Server 20.04 LTS	Apache2
lb-haproxy	e2-medium (2 vCPU, 4 GB RAM)	Ubuntu Server 20.04 LTS	HAProxy v2.x
web1	e2-micro (2 vCPU, 1 GB RAM)	Ubuntu Server 20.04 LTS	Apache2
web2	e2-micro (2 vCPU, 1 GB RAM)	Ubuntu Server 20.04 LTS	Apache2
web3	e2-micro (2 vCPU, 1 GB RAM)	Ubuntu Server 20.04 LTS	Apache2

Melalui pengaturan ini, sistem simulasi dapat dijalankan dengan efisien, memungkinkan peneliti untuk mengamati performa algoritma load balancing dalam lingkungan cloud yang mendekati kondisi nyata.

2. Instalasi Software Pendukung

Setelah seluruh virtual machine berhasil dibuat dan dikonfigurasi jaringannya, langkah selanjutnya adalah melakukan instalasi perangkat lunak pendukung yang dibutuhkan pada masing-masing VM sesuai dengan fungsinya. Pada tiga VM yang berperan sebagai web server (web1, web2, web3), dilakukan instalasi Apache HTTP server dengan perintah "sudo apt install apache2". Setelah itu, setiap web server diberikan konten sederhana berupa halaman HTML.



GAMBAR 8
HALAMAN HTML SERVER

GAMBAR 8
HALAMAN HTML SERVERGAMBAR 8

HALAMAN HTML SERVER merupakan contoh tampilan halaman HTML yang menampilkan identitas server masing-masing, sehingga memudahkan dalam mengamati distribusi beban saat pengujian. Pada VM yang berperan sebagai *load balancer,* dilakukan instalasi *HAProxy* menggunakan perintah "sudo apt install haproxy". Setelah berhasil diinstal, file konfigurasi utama HAProxy yang berada di "/etc/haproxy/ haproxy/.cfg" disesuaikan untuk masing-masing skenario algoritma, yaitu algoritma *Least Connection* dan *Weighted Round Robin.* Konfigurasi backend server dilakukan dengan mencantumkan IP internal dari masing-masing web server agar distribusi beban dilakukan secara local dalam VPC, sehingga minim latensi.

a. Hasil Percobaan

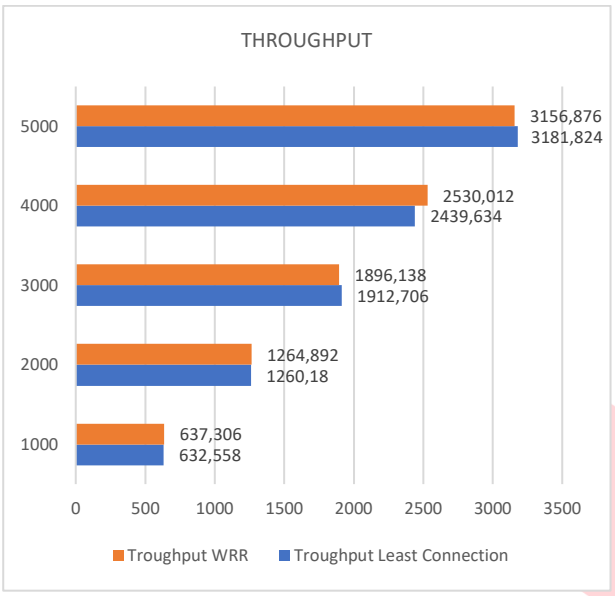
Pengujian dilakukan dengan variasi jumlah permintaan (requests) mulai dari 1000 hingga 5000 dan tingkat concurrent mulai dari 20 hingga 100. Parameter evaluasi meliputi throughput, delay dan jitter. Hasil pengujian seperti pada Tabel 4. 2 Hasil Pengujian berikut

TABEL 2
HASIL PERCOBAAN

Request	Con-current	Algoritma	Throughput (KB/s)	Delay	Jitter (ms)
1000	20	Least Connection	632,558	1.578	0,015
		WRR	637,306	1.572	0,003
2000	40	Least Connection	1260,18	0.792	0,006
		WRR	1264,892	0.789	0,0012
3000	60	Least Connection	1912,706	0.523	0,0015
		WRR	1896,138	0.522	0,001
4000	80	Least Connection	2439,634	0.428	0,027
		WRR	2530,012	0.429	0,0222
5000	100	Least Connection	3181,824	0.316	0,0035
		WRR	3156,876	0.317	0,0015

Throughput merepresentasikan jumlah data yang dapat dikirimkan dalam satuan waktu (KB/s). Untuk

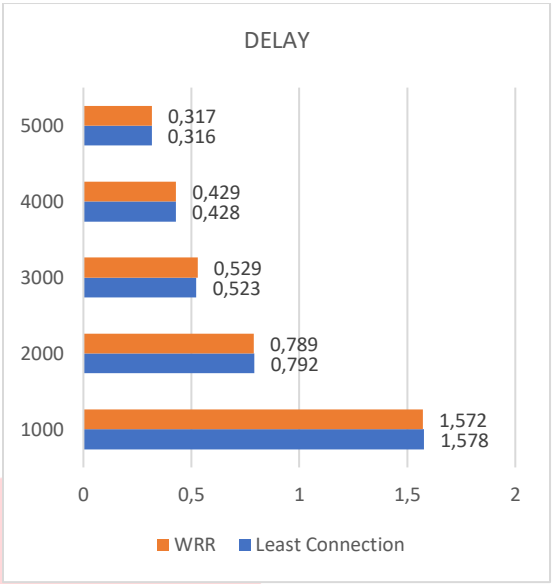
keseluruhan pengujian throughput yang sudah dilakukan, dapat dilihat hasilnya pada grafik berikut:



GAMBAR 9
THROUGHPUT

Berdasarkan hasil pengujian, pada Gambar 9 THROUGHPUT algoritma WRR menunjukkan throughput yang lebih tinggi pada jumlah request 1000–4000. Namun, pada 5000 request dengan 100 concurrent, Least Connection mampu mempertahankan throughput lebih stabil. Hal ini dapat terjadi karena WRR mengatur distribusi berdasarkan bobot server, sehingga dapat memaksimalkan potensi dari server dengan kapasitas lebih besar. Sedangkan Least Connection lebih adaptif terhadap penambahan koneksi aktif. Dengan memprioritaskan server yang memiliki koneksi aktif lebih sedikit. Least Connection mampu mencegah overload pada server tertentu dan menjaga distribusi beban tetap seimbang saat lalu lintas meningkat tajam.

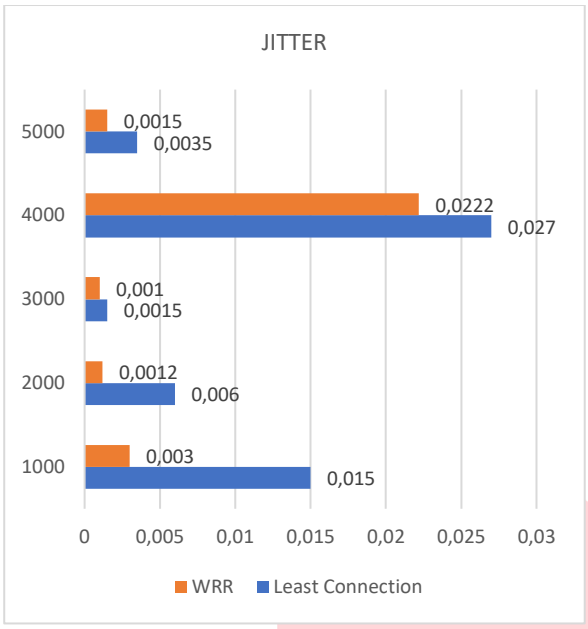
Delay dihitung berdasarkan time per request (ms). Ini adalah waktu rata-rata yang dibutuhkan untuk melayani satu permintaan dari klien. Nilai delay yang rendah menunjukkan respons sistem yang cepat. Untuk hasil pengujian delay keseluruhan dapat dilihat pada grafik berikut:



GAMBAR 10
DELAY

Berdasarkan Gambar 10 DELAY WRR secara konsisten mencatat waktu per request sedikit lebih rendah, menandakan distribusi beban yang lebih efisien dalam menangani request secara paralel sehingga menghindari penumpukan permintaan pada satu titik. Sedangkan Least Connection, meskipun sedikit lebih lambat dalam hal waktu per request, tetap menunjukkan nilai delay yang stabil dan tidak fluktuatif secara signifikan, terutama saat concurrent user bertambah. Perbedaan yang sangat tipis mengindikasikan bahwa keduanya efisien, namun WRR menunjukkan keunggulan dalam response time.

Jitter mengukur variasi delay antar permintaan. Nilai jitter yang tinggi mengidentifikasi ketidakkonsistenan sistem dalam merespons permintaan, yang tentunya sangat tidak diinginkan terutama dalam aplikasi real-time. Jitter dihitung secara manual dengan menghitung standar deviasi pada masing-masing set request-concurrent, yang mana pada penelitian ini dilakukan 5 kali percobaan pada setiap skenario. Adapun perhitungan jitter untuk keseluruhan pengujian dapat dilihat pada grafik berikut:



GAMBAR 11
JITTER

Berdasarkan Gambar 11 JITTER hasil penelitian menunjukkan bahwa kedua algoritma menghasilkan jitter yang rendah, namun WRR memiliki jitter yang sedikit lebih kecil secara rata-rata. Ini menunjukkan bahwa WRR memberikan distribusi beban yang lebih konsisten, sedangkan Least Connection cenderung mengalami variasi delay saat beban meningkat secara dinamis.

3. Analisis Hasil

TABEL 3
ANALISIS HASIL

		Least Connection	Weight Round Robin
Parameter QoS	**Throughput**	Unggul pada beban tinggi	Unggul pada beban ringan-sedang
	Delay	Cukup baik, sedikit lebih tinggi	Lebih rendah, cepat merespons
	Jitter	Stabil, sedikit fluktuatif	Lebih stabil dan konsisten
Kesesuaian GCP		Cocok untuk distribusi dinamis	Cocok untuk distribusi stabil

Berdasarkan TABEL 3 *ANALISIS HASIL* tersebut, algoritma Weighted Round Robin unggul dalam efisiensi dan kestabilan distribusi pada skenario normal, sedangkan Least connection lebih stabil pada beban tinggi dan lingkungan koneksi dinamis. Oleh karena itu, pemilihan algoritma perlu disesuaikan dengan konteks implementasi : jika sistem memiliki server dengan bobot berbeda dan trafik yang relatif stabil, WRR adalah pilihan yang tepat. Sebaliknya, jika sistem menghadapi trafik yang fluktuatif dan jumlah koneksi yang berubah cepat, Least Connection lebih disarankan

V. KESIMPULAN

Berdasarkan hasil pengujian dan analisis, kedua algoritma load balancing, yaitu Least Connection dan Weighted Round Robin, dapat diimplementasikan secara efektif di lingkungan Google Cloud Platform untuk mendistribusikan beban secara efisien. Algoritma Weighted Round Robin menunjukkan nilai delay yang sedikit lebih rendah secara konsisten di setiap skenario pengujian, menandakan efektivitasnya dalam mengatur distribusi beban berdasarkan bobot. Sementara itu, algoritma Least Connection menunjukkan performa throughput yang lebih stabil dan unggul dalam skenario beban tinggi (jumlah permintaan 5000), menunjukkan kestabilannya dalam mengatur koneksi aktif. Nilai jitter dari kedua algoritma relatif kecil dan tidak menunjukkan fluktuasi besar, sehingga keduanya layak untuk sistem yang membutuhkan kestabilan performa jaringan.

REFERENSI

[1] Desmira and D. Apriana, “View of Analisa Jaringan Local Area Network Pada Laboratorium Komputer SMK Informatika Kota Serang,” *Jurnal Inovasi dan Sains Teknik Elektro*, vol. 3, May 2022.

[2] 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, Nov. 2020, doi: 10.32832/kreatif.v8i2.3503.

[3] S. Dwiyatno, E. Rakhmat, and O. Gustiawan, “IMPLEMENTASI VIRTUALISASI SERVER BERBASIS DOCKER CONTAINER,” vol. 7, no. 2, 2020.

[4] Maji Sapdiaz, T. E. Panggabean, and I. J. Tarigan, “Building E-Learning Application Using Cloud Computing with Software As A Service (SAAS) Model,” *Antivirus : Jurnal Ilmiah Teknik Informatika*, vol. 17, no. 1, pp. 123–134, Oct. 2023, doi: 10.35457/antivirus.v17i1.3172.

[5] Alex Xu, “Common Load-Balancing Algoritms.” Accessed: May 07, 2025. [Online]. Available: https://blog.bytebytego.com/p/ep47-common-load-balancing-algorithms

[6] H. Wijaya, I. Abdurrohim, J. Tugiyono, and R. J. Rumandan, “Implementasi Metode Load Balancing Untuk Optimalisasi Performa Server Pada Jaringan Internet,” *Journal of Information System Research (JOSH)*, vol. 5, no. 1, pp. 252–260, Oct. 2023, doi: 10.47065/josh.v5i1.4386.

[7] M. Surya Pradana and A. Prapanca, “Analisis Performa Load Balancing Algoritma Weighted Round Robin di Infrastruktur BPBD Provinsi Jawa Timur,” *Journal of Informatic and Computer Science*, vol. 01, 2019.

[8] I. N. ’Abidah, M. A. Hamdani, and Y. Amrozi, “Implementasi Sistem Basis Data Cloud Computing pada Sektor Pendidikan,” *KELUWIH: Jurnal Sains dan Teknologi*, vol. 1, no. 2, pp. 77–84, Aug. 2020, doi: 10.24123/saintek.v1i2.2868.

[9] N. Ramsari and A. Ginanjar, “Implementasi Infrastruktur Server Berbasis Cloud Computing Untuk Web Service Berbasis Teknologi Google Cloud Platform,” 2022, doi: 10.28989/senatik.v7i1.472.

[10] M. Hasin and A. Zuhri, “PENENTUAN JUMLAH SERVER MENGGUNAKAN SERVER RESPONSE TIME UNTUK PENERAPAN LOAD BALANCER

PADA APLIKASI PENDAFTARAN SISWA BARU," *JOUTICA*, vol. 7, p. 561, 2022.
[11] I. Putra, Y. Prayudi, and A. Luthfi, "Live Forensics untuk mengenali Karakteristik Serangan File Upload Guna Meningkatkan Keamanan pada Web Server," 2023. [Online]. Available: http://Jiip.stkipyapisdompu.ac.id
[12] J. E. W. Prakasa, A. Hanani, F. R. Hariri, and S. N. Utama, "Improving Moodle Performance Using HAProxy and MariaDB Galera Cluster," *Applied Information System and Management (AISM)*, vol. 7, no. 1, pp. 29–36, Apr. 2024, doi: 10.15408/aism.v7i1.34871.
[13] H. Syahrizal and M. S. Jailani, "Jenis-Jenis+Penelitian+Dalam+Penelitian+Kuantitatif+dan+Kualitatif (1)," *Jurnal Pendidikan, Sosial & Humaniora*, vol. 1, May 2023.
[14] R. Kurniawan Adi Pratama, "Jurnal Restikom : Riset Teknik Informatika dan Komputer Implementasi HAProxy sebagai External Traffic Load Balancer Cluster Kubernetes yang Berjalan di atas Openstack," vol. 6, no. 3, pp. 462–473, 2024, [Online]. Available: https://restikom.nusaputra.ac.id