PERBANDINGAN PERFORMA LOAD BALANCING PADA WEB SERVER DI MICROSOFT AZURE MENGGUNAKAN ALGORITMA LEAST CONNECTION DAN ROUND ROBIN

1st Rafly Naufal Herdiansyah
Fakultas Ilmu Terapan
Universitas Telkom
Bandung, Indonesia
herdiansyahrn@student.telkomuniversity.ac.id

2nd Muhammad Iqbal
Fakultas Ilmu Terapan
Universitas Telkom
Bandung, Indonesia
miqbal@telkomuniversity.ac.id

3rd Suci Aulia
Fakultas Ilmu Terapan
Universitas Telkom
Bandung, Indonesia
suciaulia@telkomuniversity.ac.id

***Abstrak* — Seiring meningkatnya penggunaan layanan berbasis web, server sering kali mengalami lonjakan permintaan yang dapat menyebabkan keterlambatan waktu respon, penurunan performa, bahkan kegagalan layanan. Oleh karena itu, dibutuhkan mekanisme pengelolaan beban kerja server yang efisien agar performa sistem tetap optimal. Penelitian ini dilakukan sebagai solusi untuk mengatasi masalah tersebut dengan menerapkan metode load balancing menggunakan dua algoritma, yaitu *Least Connection* dan *Round Robin* pada *web server* berbasis Microsoft Azure. Penelitian ini membandingkan performa kedua algoritma berdasarkan tiga parameter utama, yaitu waktu respons, *throughput*, dan stabilitas koneksi. Pengujian dilakukan dengan mengirimkan sejumlah *request* melalui dua klien, dan pemantauan dilakukan dengan bantuan Netdata sebagai alat *monitoring* performa server. Hasil pengujian menunjukkan bahwa *algoritma Least Connection* memberikan performa lebih baik dalam kondisi beban sedang hingga tinggi karena mampu menyesuaikan beban ke server yang lebih ringan secara dinamis. Sementara itu, algoritma *Round Robin* lebih unggul pada permintaan rendah, karena mendistribusikan beban secara merata dan cepat tanpa memperhitungkan kondisi server.**

Kata kunci*— *Load Balancing, Least Connection, Round Robin, Web Server, Microsoft Azure, Netdata

I. PENDAHULUAN

Di era perkembangan teknologi digital saat ini, terjadi peningkatan signifikan pengguna layanan berbasis web yang menyebabkan lonjakan request pada server. Namun, seiring meningkatnya pengguna yang mengakses, server dapat mengalami kelebihan beban. Kondisi kelebihan beban pada gilirannya dapat menyebabkan penurunan kinerja, waktu respons yang lambat, dan bahkan kegagalan akses bagi pengguna atau yang dikenal juga dengan istilah *overload request* [1]. Perkembangan ini menuntut adanya sistem server yang andal dan efisien yang dapat menangani peningkatan lalu lintas (*traffic*) pengguna tanpa mengorbankan kualitas performa layanan. Salah satu solusi utama untuk mengatasi masalah ini adalah penggunaan mekanisme *load balancing. Load balancing* merupakan metode untuk mendistribusikan *traffic* jaringan secara merata di berbagai server, yang bertujuan untuk mengoptimalkan penggunaan sumber daya, memaksimalkan *throughput*, meminimalkan waktu respons, dan meningkatkan ketersediaan sistem secara keseluruhan [2].

Load balancing memiliki beberapa algoritma untuk menentukan cara pendistribusian beban, di antaranya adalah *Least Connection* dan *Round Robin*. Dengan algoritma *Least Connection*, permintaan baru akan diteruskan ke server dengan jumlah koneksi aktif paling sedikit. Sedangkan dengan algoritma *Round Robin*, permintaan didistribusikan dalam grup secara bergantian ke setiap server yang tersedia [2]. Dalam algoritma *Least Connection*, distribusi beban kerja dilakukan berdasarkan pemantauan kondisi aktual setiap server. Request berikutnya akan diterima oleh server dengan jumlah koneksi aktif terendah. Sementara itu, algoritma *Round Robin* bekerja dengan konsep antrian sederhana, di mana setiap request dari klien didistribusikan secara berurutan ke setiap server yang tersedia tanpa mempertimbangkan beban server saat ini.

Salah satu platform cloud terkemuka yang menyediakan layanan *load balancing* ini adalah Microsoft Azure. Layanan *Load Balancer* dirancang untuk mendistribusikan lalu lintas jaringan yang masuk ke berbagai *web server* secara merata [3]. Pentingnya mekanisme semacam ini di lingkungan cloud ditekankan oleh kebutuhan untuk mengelola sumber daya virtual secara dinamis dan efisien guna menjamin *Quality of Service* (QoS) yang konsisten bagi pengguna akhir [4]. Layanan ini mendukung berbagai algoritma untuk menentukan bagaimana permintaan diteruskan ke server, termasuk di antaranya adalah *Least Connection* dan *Round Robin*.

Berdasarkan permasalahan tersebut, penelitian ini menjadi relevan dilakukan guna melakukan perbandingan terhadap performa load balancing pada web server di Microsoft Azure menggunakan algoritma *Least Connection* dan *Round Robin*. Perbandingan ini akan difokuskan pada parameter-parameter kunci seperti waktu respons, *throughput*, dan stabilitas koneksi.

II. KAJIAN TEORI

A. Cloud Computing

Cloud Computing merupakan sebuah paradigma komputasi yang data dan programnya disimpan serta diakses melalui internet, bukan dari media penyimpanan lokal seperti

hard drive komputer. Istilah *cloud* sendiri pada dasarnya adalah metafora untuk internet [5].

B. Load Balancing

Load Balancing merupakan mekanisme yang mendistribusikan permintaan masuk ke sejumlah komputer atau kluster komputer untuk mencapai penggunaan sumber daya yang tersedia secara optimal, meningkatkan *throughput*, mengurangi waktu respons, dan mengurangi *overload* [6].

C. Docker

Menurut [7], Docker adalah teknologi virtualisasi kontainer, seperti *virtual machine* yang sangat ringan. Selain menciptakan kontainer, alur kerja pengembang juga disediakan oleh docker yang sangat membantu orang dalam menciptakan wadah dan aplikasi di dalam kontainer lalu membagikannya di antara container.

D. Web Server

Web Server adalah mesin yang menjalankan *software* atau aplikasi untuk mendistribusikan halaman web kepada pengguna, protokol untuk mentransfer atau memindahkan file yang diminta oleh pengguna melalui protokol komunikasi tertentu [8].

E. Algoritma Least Connection

Dengan menggunakan algoritma *least connection*, sistem *load balancing* dapat memfokuskan distribusi beban kerja secara dinamis sesuai dengan keadaan nyata setiap server. Oleh karena itu, algoritma ini membantu mencegah situasi di mana satu server digunakan terlalu banyak sementara yang lain memiliki kapasitas yang tidak terpakai [2].

F. Algoritma Round Robin

Proses algoritma ini dilakukan dalam urutan siklus putaran, di mana setiap server menerima sejumlah permintaan sebelum beralih ke server berikutnya dalam daftar. Metode ini memastikan bahwa beban kerja didistribusikan secara merata di antara semua server yang terlibat, sehingga mencegah server tertentu mengalami *overload* ketika server lain tidak digunakan [2].

G. Microsoft Azure

Menurut [9], Azure adalah platform dan infrastruktur populer untuk layanan *cloud*. Azure menawarkan *Software as a Service* (SaaS), *Platform as a Services* (PaaS), dan *Infrastructure as a Service* (IaaS) dengan keandalan, skalabilitas, dan infrastruktur yang hemat biaya.

H. Netdata

Menurut [10], Netdata adalah perangkat lunak pemantauan (*monitoring*) dan peringatan (*alerting*) secara *real-time*, yang arsitekturnya didasarkan pada tiga entitas utama: Agen, *Parent*, dan *Cloud*.

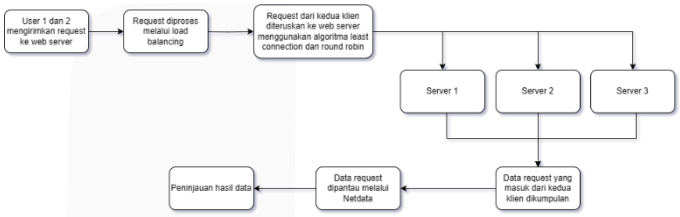
III. METODE

A. Deskripsi dan Alur Kerja Sistem Load Balancing

User 1 dan *User* 2 akan mengirim permintaan ke *web server*. *Load Balancer* bertindak sebagai penyeimbang beban yang mendistribusikan permintaan ke server yang tersedia menggunakan algoritma *Least Connection* dan *Round Robin*. Algoritma *Least Connection* akan memastikan bahwa permintaan baru dikirim ke server dengan jumlah koneksi aktif paling sedikit. Dalam mekanisme ini, contohnya Server 1 memiliki 10 koneksi, Server 2 memiliki 30 koneksi, dan Server 3 memiliki 50 koneksi. Oleh karena itu, permintaan baru akan lebih sering dikirim ke Server 1 karena memiliki jumlah koneksi aktif paling sedikit.

Sedangkan algoritma *Round Robin* menditribusikan setiap permintaan akan diarahkan ke server-server yang tersedia secara bergantian dan merata. Contohnya terdapat tiga server (Server 1, 2, dan 3) yang akan menerima dan memproses permintaan. *Load balancer* akan membagi permintaan secara berurutan ke ketiga server tersebut. Dalam mekanisme ini, ketika sembilan permintaan masuk ke sistem, pendistribusiannya akan mengikuti pola berikut: Server 1 akan menangani permintaan 1, 4, dan 7; Server 2 akan memproses permintaan 2, 5, dan 8; sementara Server 3 akan mengelola permintaan 3, 6, dan 9. Pola distribusi ini akan terus berulang untuk setiap permintaan baru yang masuk ke dalam sistem.

Setelah permintaan didistribusikan oleh *Load Balancer* ke *web server*, mekanisme berikutnya adalah *monitoring* menggunakan Netdata. Data *request* yang masuk dari kedua klien akan dikumpulkan, termasuk parameter seperti waktu respons, *throughput*, dan stabilitas koneksi. Setelah itu, dari hasil data tersebut akan ditinjau untuk membandingkan dan menentukan kelebihan dan keterbatasan dari performa kedua algoritma.



Gambar III.1 Alur Kerja Sistem Load Balancing

B. Infrastruktur yang digunakan

Berikut ini merupakan spesifikasi dari infrastruktur yang digunakan untuk pengujian performa *load balancing* dari kedua algoritma:

1. Tiga *virtual machine* yang berperan sebagai *web server*
2. Dua *virtual machine* yang akan berperan sebagai *load Balancer*
3. Satu *virtual machine* yang akan berperan sebagai klien
4. Netdata sebagai alat *monitoring* untuk keperluan pemantauan data parameter yang diukur

C. Skenario Pengujian Performa Load Balancing

Pada skenario pengujian performa *load balancing* pada web server di Microsoft azure menggunakan algoritma *least connection* dan *round robin* dengan parameter yang diukur seperti waktu respons, *throughput*, dan stabilitas koneksi. Berikut ini skenario yang telag dilakukan:

1. Skenario dengan mengirimkan 10 *request* dari dua klien sebanyak dua puluh kali
2. Skenario dengan mengirimkan 30 *request* dari dua klien sebanyak dua puluh kali

3. Skenario dengan mengirimkan 50 *request* dari dua klien sebanyak dua puluh kali
4. Skenario dengan mengirimkan 100 *request* dari dua klien sebanyak dua puluh kali
5. Skenario dengan mengirimkan 200 *request* dari dua klien sebanyak dua puluh kali
6. Skenario dengan mengirimkan 300 *request* dari dua klien sebanyak dua puluh kali

Setelah semua skenario dilakukan, maka data-data yang didapat akan dikumpulkan dan dirata-ratakan.

IV. HASIL DAN PEMBAHASAN

Hasil dari pengujian performa *load balancing* pada *web server* di microsoft azure menggunakan algoritma *least connection* dan *round robin* ini akan dibandingkan menurut parameter yang diukur seperti pada tabel-tabel berikut:

A. Tabel Perbandingan Waktu Respons

TABEL 1 WAKTU RESPONS
(A)

Load Balancer / Request	Least Connection			Round Robin		
	Avg (ms)	Min (ms)	Max (ms)	Avg (ms)	Min (ms)	Max (ms)
10	246,9	219,85	500,65	246,6	219,7	498,55
30	229,45	218,75	496,2	229	218,5	498,3
50	225,2	217,4	496,6	225,15	217,15	497,6
100	222,45	217,3	497,85	221,95	216,55	496,1
200	221,4	217,25	495,45	220,9	216,25	494,2
300	220,75	215,6	496,25	220,7	214,2	496,05

Data-data *respons time* yang dihasilkan berasal dari eksekusi (*run*) dari aplikasi apache-jmeter itu sendiri. Lalu penurunan rata-rata waktu respons bukanlah indikasi anomali, melainkan cerminan dari sistem yang telah mencapai kondisi operasional yang optimal setelah melewati fase pemanasan. Biaya inisialisasi seperti *caching* dan pembuatan koneksi sangat berpengaruh pada hasil pengujian dengan beban rendah. Sebaliknya, pada pengujian dengan beban tinggi, biaya awal ini menjadi tidak signifikan karena terdistribusi di antara ratusan *request* lain yang diproses secara efisien oleh sistem yang sudah panas.

B. Tabel Perbandingan Throughput

TABEL 2 THROUGHPUT
(A)

Load Balancer / Request	Least Connection	Round Robin
10	69 pps	83 pps
30	126 pps	106 pps
50	132.14 pps	136.79 pps
100	180.21 pps	177.54 pps
200	226.57 pps	210.68 pps
300	405 pps	381.58 pps

Berdasarkan Tabel 2, menunjukkan bahwa algoritma *least connection* cenderung memberikan performa lebih baik dibandingkan *round robin* ketika *request* meningkat. *Least connection* menunjukkan efisiensi tinggi pada beban sedang hingga tinggi yang membuatnya lebih efektif untuk digunakan pada sistem trafik tinggi.

C. Tabel Perbandingan Stabilitas Koneksi

TABEL 3 STABILITAS KONEKSI
(A)

Load Balancer / Request	Least Connection	Round Robin
10	Sedang	Tinggi
30	Tinggi	Sedang
50	Tinggi	Tinggi
100	Tinggi	Tinggi
200	Tinggi	Sedang
300	Tinggi	Sedang

Berdasarkan Tabel 3, stabilitas koneksi algoritma *Least Connection* secara konsisten berada pada tingkat tinggi, terutama saat permintaan meningkat. Hal ini disebabkan oleh kemampuannya dalam menyesuaikan beban ke server yang paling ringan secara dinamis, sehingga dapat menghindari *overload* pada server tertentu. Sementara itu, algoritma *Round Robin* menunjukkan performa stabil pada permintaan rendah hingga sedang, namun mulai kurang stabil saat beban meningkat karena tidak memperhatikan kondisi beban aktual tiap server. Hal ini menyebabkan distribusi beban menjadi kurang seimbang pada skala besar.

V. KESIMPULAN

Berdasarkan hasil dari perancangan, simulasi, serta pengujian Performa *Load Balancing* pada *Web Server* di Microsoft Azure menggunakan Algoritma *Least Connection* dan *Round Robin*, dapat diambil kesimpulan sebagai berikut:

1. Algoritma *Least Connection* dan *Round Robin* sama-sama mampu membagi beban kerja ke server dengan baik di layanan Microsoft Azure. Namun, algoritma *Least Connection* menunjukkan hasil yang lebih unggul ketika jumlah permintaan meningkat. Hal ini terbukti pada pengujian dengan 300 *request*, di mana *throughput Least Connection* mencapai 405 *packets per second* (pps), sedangkan *Round Robin* hanya 381,58 pps. Selain itu, stabilitas koneksi *Least Connection* juga tetap tinggi pada kondisi ini, sementara *Round Robin* mulai mengalami fluktuasi performa.

2. Dalam hal efisiensi pembagian beban dan kestabilan koneksi, algoritma *Least Connection* lebih unggul karena dapat mendeteksi server yang sedang tidak terlalu sibuk dan mendistribusikan permintaan secara lebih adaptif. *Round Robin* lebih cocok digunakan saat jumlah permintaan masih rendah, seperti pada pengujian dengan 10 *request*, di mana algoritma ini menunjukkan stabilitas lebih tinggi dibandingkan *Least Connection*. Hal ini terjadi karena *Round Robin* langsung membagi permintaan secara bergiliran tanpa memperhitungkan kondisi beban aktual server.
3. Kelebihan utama dari *Least Connection* adalah kemampuannya menyesuaikan pembagian beban berdasarkan kondisi nyata server, sehingga dapat menghindari *bottleneck* dan menjaga distribusi beban tetap seimbang. Namun, keterbatasannya terletak pada kebutuhan terhadap data koneksi aktif yang akurat serta proses pengambilan keputusan yang lebih kompleks. Sebaliknya, *Round Robin* memiliki keunggulan dalam kesederhanaan implementasi dan kecepatan distribusi, tetapi memiliki kelemahan karena tidak memperhitungkan beban aktual tiap server, yang menyebabkan ketidakseimbangan saat trafik tinggi, seperti terlihat pada penurunan stabilitas koneksi di pengujian dengan 200–300 *request*.
4. Berdasarkan seluruh parameter pengujian:
    a. Waktu respons kedua algoritma relatif seimbang. Contohnya, pada 300 *request*, waktu respons rata-rata *Least Connection* adalah 220,75 ms, sedangkan *Round Robin* adalah 220,7 ms.
    b. *Throughput Least Connection* unggul secara konsisten sejak 30 hingga 300 *request*.
    c. Stabilitas koneksi *Least Connection* berada pada tingkat tinggi mulai dari 30 *request* ke atas, sedangkan *Round Robin* hanya stabil hingga 100 *request*, dan mulai menurun pada 200 dan 300 *request*.

Dengan demikian, dapat disimpulkan bahwa *Least Connection* lebih optimal digunakan untuk sistem berskala besar dengan trafik dinamis dan tidak merata, karena mampu menjaga *throughput* dan kestabilan koneksi secara konsisten. Sedangkan *Round Robin* lebih sesuai untuk sistem berskala kecil hingga menengah dengan beban ringan dan merata, karena lebih sederhana dan cepat dalam proses distribusinya.

SARAN

Berdasarkan hasil penyelesaian Tugas Akhir ini, terdapat beberapa saran yang disampaikan untuk pengembangan selanjutnya, yaitu:

1. Menambahkan algoritma lain seperti *IP Hash* dan *Least Response Time* agar dapat memperoleh perbandingan performa yang lebih mendalam dan relevan dengan berbagai skenario penggunaan.
2. Menambahkan parameter pengujian seperti latensi, CPU, dan lain-lain agar hasil evaluasi performa lebih komprehensif dan menggambarkan kondisi riil.
3. Proses pemantauan performa sistem menggunakan platform monitoring yang lebih otomatis dan terintegrasi, sehingga peninjauan data dapat dilakukan secara akurat.
4. Penambahan koneksi dan *request* untuk trafik yang lebih tinggi agar dapat mendapatkan hasil performa yang lebih relevan dari algoritma-algoritma yang diuji.

REFERENSI

[1] P. A. Ghifary, S. N. Hertiana dan A. I. Irawan, "Analisis Performansi Load Balancer Menggunakan Zevenet dan Haproxy pada Tiga Web Server," *e-Proceeding of Engineering*, vol. 10, no. 4, pp. 3717-3722, 2023.

[2] D. Ariestiandy, L. Suhery, Jusmawati dan Yanuardi, "Evaluasi Load Balancing: Studi Komparatif Least-Connection dan Round-Robin dalam Konteks Cloud Computing," *JURNAL FASILKOM*, vol. 13, no. 3, pp. 424-430, Desember 2023.

[3] P. Borra, "Exploring Microsoft Azure's Cloud Computing: A Comprehensive Assessment," *International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)*, vol. 2, no. 8, pp. 897-906, Mei 2022.

[4] S. D. Riskiono dan D. Darwis, "Peran Load Balancing Dalam Meningkatkan Kinerja Web Server di Lingkungan Cloud," *KREA-TIF: JURNAL TEKNIK INFORMATIKA*, vol. 8, no. 2, pp. 1-8, November 2020.

[5] A. Rashid dan A. Chaturvedi, "Cloud Computing Characteristics and Services: A Brief Review," *International Journal of Computer Sciences and Engineering*, vol. 7, no. 2, pp. 421-426, Februari 2019.

[6] Alimuddin dan A. Ashari, "Peningkatan Kinerja Siakad Menggunakan Metode Load Balancing dan Fault Tolerance Di Jaringan Kampus Universitas Halu Oleo," *IJCCS (Indonesian Journal of Computing and Cybernetics Systems)*, vol. 10, no. 1, pp. 11-22, Januari 2016.

[7] C. Anderson, "Docker [software engineering]," *IEEE Software*, vol. 32, no. 3, pp. 102-105, 2015.

[8] A. Tedyyana dan R. Kurniati, "MEMBUAT WEB SERVER MENGGUNAKAN DINAMIC DOMAIN NAME SYSTEM PADA IP DINAMIS," *Jurnal Teknologi Informasi & Komunikasi Digital Zone*, vol. 7, no. 1, pp. 1-10, Februari 2016.

[9] R. M. H. Al-Sayyed, W. A. Hijawi, A. M. Bashiti, I. AlJarah, N. Obeid dan O. Y. Adwan, "An Investigation of Microsoft Azure and Amazon Web Services from Users' Perspectives," *International Journal of Emerging Technologies in Learning (iJET)*, vol. 14, no. 10, pp. 217-241, 2019.

[10] F. Voutsas, J. Violos dan A. Leivadeas, "Mitigating Alert Fatigue in Cloud Monitoring Systems: A Machine Learning Perspective," *Computer Networks*, no. 250, p. 11054, 2024.

[11] T. W. Harjanti, H. Setiyani dan J. Trianto, "Load Balancing Analysis Using Round-Robin and Least-

Connection Algorithms for Server Service Response Time," *Applied Technology and Computing Science Journal (ATCSJ)*, vol. 5, no. 2, pp. 119-128, Januari 2023.

[12] P. Mell dan T. Grance, "The NIST Definition of Cloud Computing," *National Institute of Standards and Technology (NIST)*, September 2011.