Journal of Renewable Energy and Smart Device (JORESD)

ISSN: 3026-216X
Vol. 3, No. 1, May 2025

Analisis Performa *Load balancing* pada Web *Server* Menggunakan Nginx

Khaidir[1] , Kholik Prasojo[2*], Herlina[3], A. Muhammad Fuad Fadhlurrahman[4]

[1,2,3,4]Universitas Negeri Makassar
[(*)]Coresponding Author E-mail: kholik.prasojo@unm.ac.id

ARTICLE HISTORY

Received :
Revised :
Accepted :

ABSTRACT

The rapid development of information technology, particularly in the use of web-based services, has resulted in increased data traffic to *server*s. Without proper management, this may lead to decreased system performance. *Load balancing* is an approach introduced in technology education to teach technical solutions for *server* load management. This study aims to analyze the impact of implementing *load balancing* using Nginx software on web *server* performance. The method applied is a quantitative experiment involving two testing scenarios: before and after the implementation of *load balancing* using the Round Robin algorithm. The testing environment was built using virtualization with the Ubuntu *Server* operating system, and performance measurements were conducted using Apache Benchmark and JMeter. The analyzed parameters include latency, throughput, and response time. The findings show that implementing *load balancing* can significantly improve system efficiency, making it a relevant learning case in computer networking and system administration courses..

Keywords: Load balancing, Nginx, web *server* performance, virtualization

1. PENDAHULUAN

Seiring dengan meningkatnya kebutuhan akan layanan web yang cepat dan andal, pengelolaan beban kerja pada *server* menjadi tantangan utama dalam infrastruktur teknologi informasi. Lonjakan jumlah pengguna dan permintaan akses yang tinggi dapat menyebabkan *server* mengalami kelebihan beban (*overload*), yang berdampak pada penurunan performa hingga kegagalan layanan[1]. Untuk mengatasi permasalahan ini, implementasi *load balancing* menjadi solusi efektif dalam mendistribusikan lalu lintas jaringan secara merata ke beberapa *server backend* [2].

Nginx merupakan salah satu perangkat lunak *open-source* yang populer digunakan sebagai *web server* sekaligus *load balancer* [3]. Kemampuannya dalam menangani koneksi simultan yang tinggi dan konfigurasi yang fleksibel menjadikannya pilihan utama dalam arsitektur sistem yang membutuhkan skalabilitas dan keandalan tinggi. Nginx mendukung berbagai algoritma distribusi beban seperti *Round Robin*, *Least Connections*, dan IP *Hash*, yang masing-masing memiliki karakteristik dan performa berbeda tergantung pada skenario penggunaan. Beberapa penelitian telah dilakukan untuk menganalisis performa *load balancing* menggunakan Nginx. Misalnya, studi oleh Adnan menunjukkan bahwa implementasi Nginx sebagai *load balancer* dapat meningkatkan stabilitas dan menjaga keseimbangan *web server*, terutama saat menggunakan algoritma *Least Connections* yang menunjukkan *response time* stabil sebesar

116ms dan throughput mencapai 2300.96 *request* per detik[2]. Penelitian lain mengungkapkan bahwa algoritma *Round Robin* memberikan *response time* tercepat sebesar 36ms dengan *throughput* 223.3 *request* per detik dalam pengujian dengan 40.000 pengguna[4]. Namun, efektivitas dari setiap algoritma *load balancing* dapat bervariasi tergantung pada konfigurasi sistem, jumlah pengguna, dan jenis aplikasi yang dijalankan. Oleh karena itu, analisis mendalam terhadap performa masing-masing algoritma dalam berbagai skenario menjadi penting untuk menentukan konfigurasi optimal yang dapat meningkatkan kinerja dan keandalan layanan *web* oleh[4].

Penelitian ini bertujuan untuk menganalisis performa *load balancing* pada *web server* menggunakan Nginx dengan membandingkan beberapa algoritma distribusi beban. Evaluasi akan dilakukan berdasarkan metrik performa seperti *response time*, *throughput*, dan penggunaan sumber daya sistem, guna memberikan rekomendasi konfigurasi terbaik dalam implementasi *load balancing* menggunakan Nginx. Meningkatnya ketergantungan masyarakat dan organisasi terhadap layanan digital menyebabkan kualitas layanan web menjadi faktor kunci dalam keberhasilan sistem informasi. Kualitas tersebut tidak hanya ditentukan oleh desain antarmuka atau konten, tetapi juga oleh performa teknis sistem seperti waktu respons, keandalan akses, dan kemampuan menangani trafik tinggi secara simultan oleh[4].

Dalam konteks ini, *load balancing* berperan penting dalam menjamin kelangsungan layanan, terutama pada sistem web berskala besar dan kompleks. *Load balancing* memungkinkan pengelolaan lalu lintas jaringan agar *server* tidak mengalami *bottleneck* yang dapat menyebabkan *downtime*. Dengan sistem ini, beban kerja didistribusikan ke beberapa *node server*, menciptakan efisiensi kerja dan redundansi layanan yang penting untuk *high availability system*.

Meningkatnya ketergantungan masyarakat dan organisasi terhadap layanan digital menyebabkan kualitas layanan web menjadi faktor kunci dalam keberhasilan sistem informasi. Kualitas tersebut tidak hanya ditentukan oleh desain antarmuka atau konten, tetapi juga oleh performa teknis sistem seperti waktu respons, keandalan akses, dan kemampuan menangani trafik tinggi secara simultan.

Dalam konteks ini, *load balancing* berperan penting dalam menjamin kelangsungan layanan, terutama pada sistem web berskala besar dan kompleks. *Load balancing* memungkinkan pengelolaan lalu lintas jaringan agar *server* tidak mengalami *bottleneck* yang dapat menyebabkan *downtime*. Dengan sistem ini, beban kerja didistribusikan ke beberapa *node server*, menciptakan efisiensi kerja dan redundansi layanan yang penting untuk *high availability system*.

2. Metodologi

Penelitian ini menggunakan pendekatan kuantitatif dengan metode eksperimen untuk menganalisis performa *load balancing* pada web *server* menggunakan Nginx. Pendekatan ini dipilih karena memungkinkan dilakukannya pengukuran secara langsung dan objektif terhadap kinerja sistem berdasarkan parameter teknis yang relevan. Eksperimen dilakukan dengan membandingkan tiga algoritma *load balancing* yang didukung oleh Nginx, yaitu *Round Robin*, *Least Connections*, dan IP *Hash*. Pengujian dilakukan dalam lingkungan virtual dengan konfigurasi topologi yang terdiri dari satu *server* sebagai *load balancer* dan tiga *server backend* yang menjalankan layanan web dengan konten seragam,seluruh *node* menggunakan sistem

operasi Ubuntu *Server* 20.04 LTS, dengan spesifikasi minimum CPU 2 core, RAM 4 GB, dan media penyimpanan 40 GB. Nginx dikonfigurasi pada *server* utama untuk mendistribusikan permintaan klien ke *server backend* berdasarkan algoritma yang diuji. Untuk mengukur performa, digunakan Apache JMeter sebagai alat uji beban, dengan skenario pengujian mencakup 50, 100, dan 200 pengguna simultan dalam durasi waktu 10 menit untuk setiap algoritma. Selama pengujian, data dikumpulkan mencakup metrik utama seperti rata-rata waktu respons, *throughput*, tingkat kesalahan, serta penggunaan CPU dan memori pada *server*. Data yang diperoleh dianalisis secara kuantitatif dengan teknik statistik deskriptif, yaitu menghitung nilai rata-rata dan membandingkan hasil dari masing- masing algoritma untuk menentukan metode distribusi beban yang paling optimal. Hasil analisis disajikan dalam bentuk tabel dan grafik untuk memudahkan interpretasi, serta digunakan sebagai dasar untuk menarik kesimpulan mengenai performa sistem berdasarkan algoritma *load balancing* yang diterapkan.

2.1 Konsep *Load Balancing*

Load balancing merupakan teknik penting dalam rekayasa jaringan dan sistem terdistribusi yang bertujuan untuk mendistribusikan lalu lintas atau permintaan layanan secara merata ke beberapa *server* atau sumber daya *backend*[5]. Pendekatan ini tidak hanya meningkatkan efisiensi penggunaan sumber daya, tetapi juga meningkatkan ketersediaan (*availability*), skalabilitas (*scalability*), dan keandalan (*reliability*) layanan[6]. Tanpa adanya *load balancing*, sistem menjadi rentan terhadap kelebihan beban pada satu titik yang dapat menyebabkan keterlambatan, penurunan performa, bahkan kegagalan layanan. Dalam konteks komputasi awan (*cloud computing*), *load balancing* menjadi elemen krusial dalam menjaga *Quality of Service* (QoS).

2.2 Nginx

Nginx adalah *server* web *open-source* yang populer digunakan sebagai *reverse proxy*, *load balancer*, dan HTTP *cache*. Didesain oleh Igor Sysoev pada tahun 2004 untuk mengatasi masalah *concurrency* dan performa rendah dari *server* tradisional seperti Apache, Nginx kini telah banyak digunakan dalam sistem berskala besar[7]. Dengan arsitektur berbasis *event-driven* dan *asynchronous*, Nginx mampu menangani ribuan koneksi secara simultan dengan konsumsi sumber daya yang rendah. Selain berfungsi sebagai *server* web, Nginx juga mendukung *load balancing* pada tingkat aplikasi dan *transport.*

2.3 Pengaruh *Load balancing* terhadap Metrik Performa *Server*

Penerapan *load balancing* pada sistem web *server* memiliki peran penting dalam peningkatan kinerja layanan digital, terutama ketika jumlah permintaan dari pengguna meningkat secara signifikan. Salah satu manfaat langsung yang dapat diamati adalah efisiensi distribusi permintaan, yang berdampak pada stabilitas waktu respons *server*. Dalam konteks pengujian ini, penggunaan Nginx sebagai *load balancer* dengan algoritma Round Robin terbukti mampu menyeimbangkan beban secara proporsional ke setiap node *server* backend. Hal ini membuat proses penanganan permintaan menjadi lebih ringan per *server* karena tidak ada titik tunggal yang menjadi beban utama.

Dari hasil pengujian yang dilakukan, terlihat adanya perbedaan nilai requests per second yang menunjukkan kapasitas masing-masing *server* dalam melayani permintaan. Walaupun

algoritma Round Robin secara teori membagi trafik secara adil, performa aktual *server* tetap dipengaruhi oleh sumber daya dan kondisi runtime dari masing- masing mesin virtual. Penggunaan *load balancing* membuat proses permintaan tidak hanya lebih terorganisir, tetapi juga membantu mencegah kemacetan pada salah satu titik layanan, yang jika dibiarkan dapat menurunkan kualitas keseluruhan sistem[8].

Selanjutnya, parameter time per request dan throughput juga mengalami peningkatan performa. Dengan adanya pembagian beban, waktu rata-rata yang dibutuhkan untuk merespons satu permintaan menjadi lebih singkat dibandingkan saat *server* bekerja secara tunggal. Hal ini menunjukkan bahwa *load balancing* tidak hanya berpengaruh pada aspek kuantitas permintaan yang dapat ditangani, tetapi juga mempercepat proses eksekusi di sisi pengguna. Efek ini sangat bermanfaat untuk aplikasi web yang membutuhkan akses cepat dan responsif.

Stabilitas sistem juga menjadi aspek penting yang terpengaruh secara positif[9]. Dalam pengujian ini, tidak ditemukan adanya permintaan yang gagal atau error koneksi, yang menandakan bahwa sistem berjalan dengan lancar dan dapat mempertahankan kualitas layanan. Dengan adanya *load balancer*, beban trafik dapat disalurkan ke *server* yang siap melayani, sehingga potensi terjadinya penurunan performa karena overload dapat diminimalkan.

Secara keseluruhan, penerapan *load balancing* menggunakan Nginx mampu memberikan dampak yang signifikan terhadap peningkatan metrik performa web *server*. Performa sistem menjadi lebih stabil, responsif, dan efisien dalam menangani trafik yang masuk. Dengan begitu, penggunaan *load balancer* bukan hanya sebagai solusi teknis, tetapi juga sebagai strategi penting dalam perancangan sistem layanan berbasis web yang andal dan skalabel.

Temuan dari penelitian ini juga sejalan dengan beberapa hasil studi terdahulu. Mulyadi dalam penelitiannya menunjukkan bahwa implementasi *load balancing* menggunakan Nginx mampu meningkatkan *throughput* hingga 25% dan mengurangi rata-rata *response time server* secara signifikan[10]. Hal ini menunjukkan bahwa *load balancing* tidak hanya membantu membagi beban, tetapi juga mempercepat eksekusi permintaan.

Selain itu, Penelitian lain mengemukakan bahwa algoritma *Round Robin* sangat efektif dalam mendistribusikan trafik secara merata, namun kurang cocok apabila sistem membutuhkan *session persistence*, di mana IP *Hash* lebih tepat digunakan[11]. Sementara itu, Prakoso dan Purnama membandingkan berbagai algoritma dan menyimpulkan bahwa pemilihan metode *load balancing* harus mempertimbangkan pola trafik pengguna serta beban *server* yang dinamis[12]. Dengan demikian, penerapan algoritma *Round Robin* dalam penelitian ini dinilai sudah tepat untuk skenario sistem yang memiliki beban seimbang dan permintaan yang tidak terikat sesi pengguna tertentu.

Selain itu, Prakoso dan Purnama menyarankan pemilihan algoritma *load balancing* harus mempertimbangkan pola trafik dan karakteristik beban sistem. Hal ini penting untuk menjaga efisiensi distribusi beban dalam berbagai skenario operasional.

Temuan Ma dan Chi memperkuat relevansi pemilihan algoritma dengan membandingkan algoritma *default* Nginx dengan algoritma baru bernama NEW_HASH. Algoritma ini terbukti lebih optimal dalam meningkatkan throughput dan mengurangi waktu respons dibandingkan IP_HASH standar[13]. Sementara itu, Hu mengusulkan pendekatan *load balancing* adaptif berbasis algoritma genetika untuk menyesuaikan distribusi beban secara dinamis berdasarkan

metrik seperti CPU, memori, dan I/O. Hasilnya menunjukkan peningkatan performa sistem sebesar 15% dibandingkan metode statis[14].

Tak kalah penting, studi oleh Putro dan Supono menemukan bahwa Nginx memiliki keunggulan dibandingkan Apache dalam menangani beban tinggi, baik dari sisi distribusi trafik maupun waktu respons. Ini menegaskan bahwa pemilihan teknologi dan algoritma yang sesuai sangat berpengaruh dalam desain arsitektur sistem yang skalabel dan andal[15].

3. HASIL DAN PEMBAHASAN

3.1 Konfigurasi *Load Balance*

Load Balancer merupakan komponen utama dalam sistem terdistribusi ini yang berfungsi untuk mendistribusikan permintaan (*request*) dari klien ke beberapa Web *Server* secara merata. Dalam konfigurasi ini, metode distribusi yang digunakan adalah *round-robin*, yaitu metode yang membagikan trafik secara bergiliran ke setiap *server* yang tersedia dalam daftar backend. Pendekatan ini bertujuan untuk memastikan bahwa tidak ada satu *server* pun yang terbebani secara berlebihan.

Langkah-langkah konfigurasi *Load Balancer* yang dilakukan adalah sebagai berikut

a) Menentukan IP *Address Web Server* Masing-masing Web *Server* memiliki alamat IP statis yang digunakan sebagai identitas dalam konfigurasi backend. Misalnya, Web Server 1 memiliki IP 27.0.1.1, sedangkan Web *Server* 2 memiliki IP 27.0.0.1.
b) Mengatur File Konfigurasi *Load Balancer* Penulis menggunakan aplikasi *proxy* seperti HAProxy atau Nginx sebagai *Load Balancer*. Konfigurasi *backend* dituliskan dalam file konfigurasi utama, dengan mendefinisikan dua *server* tujuan dan strategi *load balancing*.
c) Menjalankan dan Menguji *Load Balancer* Setelah konfigurasi selesai, layanan *Load Balancer* dijalankan. Pengujian dilakukan dengan mengakses IP *Load Balancer* dari *browser* dan mengamati hasil respon yang diterima. terlihat bahwa halaman web yang muncul berasal secara bergantian dari Web *Server* 1 dan Web *Server* 2. Hal ini menandakan bahwa *Load Balancer* bekerja dengan baik dalam mengarahkan trafik.
d) Verifikasi Melalui Terminal Pengujian lanjutan juga dilakukan menggunakan perintah *curl* dari terminal untuk memastikan respon HTTP berasal dari dua *server* yang berbeda secara konsisten.

Secara keseluruhan, konfigurasi *Load Balancer* berhasil dilakukan tanpa kendala yang berarti. Sistem dapat membagi beban secara merata, sehingga meningkatkan efisiensi dan reliabilitas layanan.

3.2 Konfigurasi Web *Server 1* dan Web *Server* 2

Web *Server* merupakan komponen yang bertugas menyediakan konten atau layanan kepada pengguna akhir. Dalam implementasi sistem ini, dua buah Web *Server* digunakan secara bersamaan untuk melayani permintaan yang diteruskan oleh *Load Balancer*. Masing-masing Web *Server* dikonfigurasi secara terpisah namun memiliki fungsi yang setara, yaitu untuk menanggapi permintaan HTTP dan menampilkan halaman web sederhana sebagai bentuk identifikasi *server*.

Berikut ini adalah langkah-langkah konfigurasi yang dilakukan pada kedua Web *Server*:

a) Instalasi Web *Server*
   Pada masing-masing *server*, dilakukan instalasi perangkat lunak web *server* seperti Apache (apache2) atau Nginx menggunakan manajer paket bawaan sistem operasi.
b) Pemeriksaan Status dan Pengaktifan Layanan Setelah proses instalasi selesai, layanan web *server* dijalankan dan diatur agar aktif secara otomatis saat sistem menyala:
c) Pembuatan Halaman Web untuk Identifikasi Untuk membedakan antara Web *Server* 1 dan Web *Server* 2, masing-masing *server* diberi halaman HTML sederhana dengan isi yang berbeda. Misalnya:
   1) Web *Server* 1, menampilkan tulisan "Ini adalah Web *Server* 1".
   2) Web *Server* 2, menampilkan tulisan "Ini adalah Web *Server* 2". File tersebut disimpan di direktori *root web server*, seperti /var/www/html/index.html.
d) Pengujian Akses Lokal
   Setelah halaman dibuat, dilakukan pengujian menggunakan *browser* atau *curl* dengan mengakses alamat IP masing-masing *server* secara langsung.
e) Konektivitas ke *Load Balancer*
   Setelah kedua Web *Server* berhasil memberikan respon melalui akses langsung, dilakukan pengujian dengan mengakses alamat IP *Load Balancer*. Hasilnya, tampak bahwa halaman dari kedua *server* dapat ditampilkan secara bergantian, menandakan bahwa Web *Server* sudah terkoneksi dan dikontrol dengan baik oleh *Load Balancer*.

3.3 Pengujian Sistem

Pengujian sistem merupakan tahap penting untuk memastikan bahwa konfigurasi *Load Balancer* dan kedua Web *Server* telah berjalan sesuai dengan tujuan yang dirancang. Tujuan utama dari pengujian ini adalah untuk melihat apakah *Load Balancer* benar-benar mampu mendistribusikan permintaan dari klien secara merata ke dua *server backend*, serta memastikan bahwa masing-masing Web *Server* dapat memberikan respon dengan benar.

Langkah-langkah pengujian dilakukan sebagai berikut:

a) Pengujian Akses Melalui *Load Balancer*
   Pengguna mengakses alamat IP publik atau lokal dari *Load Balancer* melalui *browser*. Jika konfigurasi berhasil, maka halaman web yang muncul akan bergantian menampilkan konten dari Web *Server* 1 dan Web *Server* 2. Hal ini terjadi karena *Load Balancer* mengarahkan permintaan secara bergiliran ke masing-masing *server* berdasarkan algoritma *round-robin*.
b) Pengujian Berulang (*Refresh Page*)
   Untuk memastikan sistem *load balancing* berfungsi dengan baik, halaman web di-*refresh* secara terus menerus. Dari hasil pengujian, setiap kali halaman di-*refresh*, konten berganti antara "Ini adalah Web *Server* 1" dan "Ini adalah Web *Server* 2". Ini membuktikan bahwa *Load Balancer* berhasil membagi trafik dengan adil.
c) Pengujian Melalui Terminal (*curl*)
   Selain menggunakan *browser*, pengujian juga dilakukan menggunakan perintah *curl* dari terminal. Tujuannya adalah untuk melihat respon HTTP secara langsung dan memastikan bahwa isi halaman berasal dari dua sumber yang berbeda. Perintah yang digunakan:

curl http://<IP-Load-Balancer>

Hasil yang ditampilkan menunjukkan isi halaman yang berbeda secara bergantian, sesuai dengan konfigurasi kedua Web *Server*.

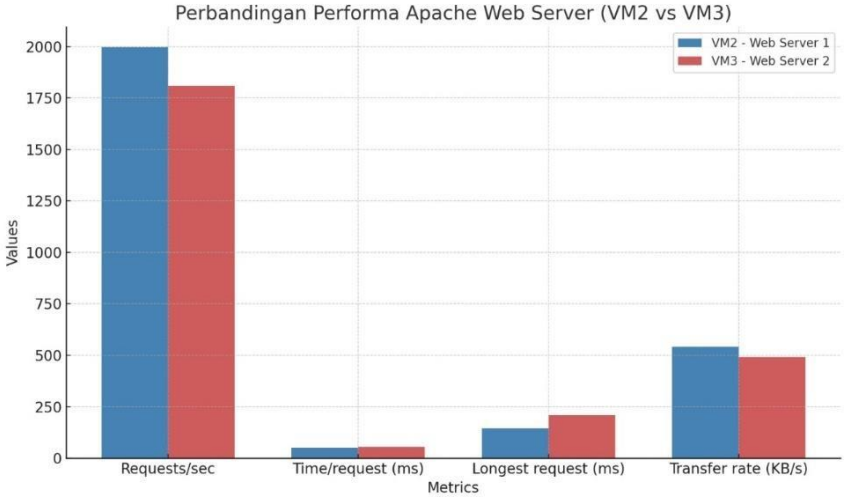
d) Pengujian Stabilitas Sistem

Pengujian dilakukan dalam jangka waktu tertentu dengan beban ringan. Selama proses ini tidak ditemukan kendala seperti keterlambatan respon (*latency* tinggi) atau error koneksi, yang menandakan bahwa sistem telah berjalan secara stabil dalam skenario pengujian dasar.

Pengujian dilakukan untuk membandingkan performa web *server* 1 VM2) dan web *server* 2 VM3 menggunakan ApacheBench. Berikut adalah rangkuman hasil pengujian:

Tabel 1. Uji Performa Web Server 1 dan Web Server 2

Parameter	**Web *Server* 1 (VM2)**	**Web *Server* 2 (VM3)**
Requests per second	1996.39 req/sec	1808.48 req/sec
Time per request (mean)	50.090 ms	55.295 ms
Longest request	144 ms	209 ms
Transfer rate	541.99 KB/sec	490.97 KB/sec
Failed requests	0	0
Total transferred	278000 bytes	278000 bytes
HTML transferred	32000 bytes	32000 bytes
Concurrency level	100	100



Gambar 1. Matriks Performa Web Server Uji

Web *Server* 1 (VM2) menunjukkan performa yang lebih baik dibandingkan Web *Server* 2 (VM3) dalam semua metrik kunci, yang mengindikasikan bahwa VM2 mampu menangani lebih banyak permintaan dengan waktu respons yang lebih cepat serta kecepatan transfer yang lebih tinggi. Meskipun terdapat perbedaan performa, kedua *server* menunjukkan tingkat stabilitas layanan yang baik karena tidak ada permintaan yang gagal selama pengujian.

3.4 Analisis Hasil Pengujian

Perbedaan performa antara Web *Server* 1 (VM2) dan Web *Server* 2 (VM3) dalam hasil pengujian mengindikasikan bahwa meskipun algoritma *Round Robin* membagi trafik secara merata, tetap ada variabel lain yang memengaruhi kualitas respons dari masing-masing *server*. Kemungkinan seperti perbedaan alokasi sumber daya virtual, proses latar belakang, atau keterbatasan I/O *storage* dapat menjadi faktor yang memengaruhi performa akhir *server*.

Penerapan algoritma *Round Robin* dalam studi ini memberikan gambaran bahwa metode ini bekerja efektif dalam skenario distribusi sederhana, di mana permintaan dibagi rata ke seluruh node tanpa mempertimbangkan beban yang sedang berlangsung. Namun, kelemahan dari metode ini adalah kurang responsif terhadap kondisi nyata sistem yang dinamis, karena tidak mempertimbangkan jumlah koneksi aktif atau beban kerja aktual di masing-masing *server backend*. Berdasarkan hasil pengujian, sistem dapat dikatakan berjalan stabil karena tidak ditemukan adanya *failed request*, dan kedua server mampu merespon permintaan dengan baik. Hal ini menunjukkan bahwa dari sisi arsitektur, sistem telah berhasil dirancang dengan benar. Meskipun demikian, jika penelitian ini diperluas untuk skala pengguna yang lebih besar atau waktu pengujian yang lebih panjang, perbedaan performa antar *server* bisa menjadi lebih mencolok dan memengaruhi kualitas layanan secara keseluruhan.

Dengan demikian, dapat disimpulkan bahwa algoritma *Round Robin* cocok digunakan pada lingkungan pembelajaran atau produksi berskala kecil yang memiliki *server* dengan spesifikasi seragam. Untuk kebutuhan produksi skala besar, akan lebih baik mempertimbangkan algoritma adaptif seperti *Least Connections* atau *Weighted Round Robin* agar beban dapat didistribusikan lebih adil berdasarkan kapasitas dan kondisi terkini *server*. Akhirnya, penelitian ini memberikan pemahaman praktis yang bermanfaat bagi mahasiswa dan praktisi, khususnya dalam merancang sistem yang skalabel dan efisien.

Implikasi dari hasil ini membuka peluang pengembangan lebih lanjut dalam skenario kompleks seperti penambahan *health check*, *failover* otomatis, dan integrasi dengan sistem pemantauan performa *real-time*.

4. KESIMPULAN DAN SARAN

Berdasarkan hasil konfigurasi dan pengujian yang telah dilakukan, dapat disimpulkan bahwa penerapan load balancing menggunakan Nginx memberikan dampak positif terhadap peningkatan performa layanan web server. Sistem yang dibangun mampu mendistribusikan permintaan secara merata ke dua Web Server backend menggunakan algoritma round-robin, yang terbukti efektif dalam menjaga kestabilan dan efisiensi sistem.

Hasil pengujian menunjukkan bahwa Load Balancer berhasil mengatur lalu lintas dengan adil dan setiap permintaan dari klien mendapat respon dari server yang berbeda secara bergiliran. Selain itu, konfigurasi Web Server yang konsisten serta keterhubungan yang baik dengan Load Balancer menunjukkan bahwa sistem ini berjalan tanpa adanya hambatan berarti.

Implementasi ini dapat menjadi studi kasus yang aplikatif dalam pembelajaran jaringan komputer dan sistem terdistribusi, karena mencerminkan praktik nyata dalam dunia industri untuk mengelola trafik yang tinggi secara efisien.

REFERENSI

[1] P. Dymora, M. Mazurek, and B. Sudek, "Comparative analysis of selected open-source solutions for traffic balancing in server infrastructures providing www service," *Energies (Basel)*, vol. 14, no. 22, Nov. 2021, doi: 10.3390/en14227719.

[2] F. Apriliansyah, I. Fitri, A. Iskandar, and R. Artikel, "Jurnal Teknologi dan Manajemen Informatika Implementasi Load Balancing Pada Web Server Menggunakan Nginx Info Artikel ABSTRAK," *Tahun*, vol. 6, no. 1, 2020, [Online]. Available: http://http://jurnal.unmer.ac.id/index.php/jtmi

[3] B. Setyaji, M. Pranata, and I. Kresna, "PERFORMANCE ANALYSIS OF LOAD BALANCING LEARNING MANAGEMENT SYSTEM MOODLE ON DOCKER CONTAINER," 2023.

[4] M. Rafli, I. Fitri, and A. Andrianingsih, "Pengujian Kinerja Load Balancing Web Server Menggunakan Nginx Riverse Proxy Berbasis OS Centos 7," vol. 9, no. 3, pp. 1824–1840, 2022, [Online]. Available: http://jurnal.mdp.ac.id

[5] M. H. Kashani, A. Ahmadzadeh, and E. Mahdipour, "Load balancing mechanisms in fog computing: A systematic review."

[6] J. Wang and Z. Kai, "Performance Analysis and Optimization of Nginx-based Web Server," in *Journal of Physics: Conference Series*, IOP Publishing Ltd, Jun. 2021. doi: 10.1088/1742-6596/1955/1/012033.

[7] E. Danilevičius and L. Kaklauskas, "STUDY OF HIGH AVAILABILITY AND PERFORMACE OFF SERVER CLUSTER."

[8] S. K. Saeid and T. A. Yahiya, "Load Balancing Evaluation Tools for a Private Cloud: A Comparative Study," 2018, doi: 10.14500/aro.10438.

[9] M. A. Shahid, M. M. Alam, and M. M. Su'ud, "Performance Evaluation of Load-Balancing Algorithms with Different Service Broker Policies for Cloud Computing," *Applied Sciences (Switzerland)*, vol. 13, no. 3, Feb. 2023, doi: 10.3390/app13031586.

[10] "3792-Research Instrument-16263-15294-10-20200916".

[11] L. H. Pramono, R. C. Buwono, and Y. G. Waskito, "Round-robin algorithm in HAProxy and nginx load balancing performance evaluation: A review," in *2018 International Seminar on Research of Information Technology and Intelligent Systems, ISRITI 2018*, Institute of Electrical and Electronics Engineers Inc., Nov. 2018, pp. 367–372. doi: 10.1109/ISRITI.2018.8864455.

[12] M. Ilham, "Implementasi Sistem Load Balancing untuk Optimasi Kinerja pada Web Server Nginx Menggunakan Algoritma Ip Hash."

[13] C. Ma and Y. Chi, "Evaluation Test and Improvement of Load Balancing Algorithms of Nginx," *IEEE Access*, vol. 10, pp. 14311–14324, 2022, doi: 10.1109/ACCESS.2022.3146422.

[14] Y. Hu, Z. Bao, and X. Bao, "A Dynamic–Static Load Balancing Algorithm Based on Improved Genetic Algorithm," *Electronic Science & Technology*, vol. 36, no. 9, Sep. 2023, doi: 10.16180/j.cnki.issn1007-7820.2023.09.012.

[15] Z. P. Putro and R. A. Supono, "Comparison Analysis of Apache and Nginx Webserver Load Balancing on Proxmox VE in Supporting Server Performance," *International Research Journal of Advanced Engineering and Science*, vol. 7, no. 3, pp. 144–151, 2022.