Krea-TIF: Jurnal Teknik Informatika
http://ejournal.uika-bogor.ac.id/index.php/krea-tif
p-ISSN: 2338-2910, e-ISSN: 2658-5836
Vol. 12, No. 1, 2024, pp. 24 – 31
DOI: 10.32832/krea-tif.v12i1.15964

Analisis Kinerja *Load Balancing* dengan Metode *Source Hash Scheduling* dan URI pada Web Server

Zulhipni Reno Saputra Elsi[1], Dedy Alamsyah[2], Jusmawati[3], Nurhayati[4]
[1]Program Studi Teknologi Informasi, Universitas Muhammadiyah Palembang, Indonesia
[2,4]Program Studi Teknik Informatika, Universitas Muhammadiyah Tangerang, Indonesia
[3]Program Studi Sistem Informasi, Universitas Yapis Papua, Indonesia
E-mail koresponden: zulhipni_renosaputra@um-palembang.ac.id

Diserahkan *21 Januari 2024;* ***Direview*** *25 Mei 2024;* ***Dipublikasikan*** *30 Mei 2024*

Abstrak

Pesatnya pertumbuhan internet saat ini, terjadi peningkatan signifikan dalam akses pengguna yang terkoneksi. Fenomena ini berdampak langsung pada kebutuhan mesin penyedia layanan, terutama pada server web. Penelitian ini bertujuan untuk menganalisis kinerja web server dengan menerapkan strategi load balancing menggunakan dua algoritma berbeda, yaitu Source Hash Scheduling dan Uniform Resource Identifier (URI). Load balancing diuji untuk menentukan efektivitasnya dalam mengoptimalkan response time. Data yang dikumpulkan dari pengujian menunjukkan hasil rata-rata response time dalam milidetik untuk berbagai tingkat koneksi, mulai dari 100/100 hingga 5000/500. Hasil pengujian menunjukkan bahwa pada tingkat koneksi 1000/100, algoritma Source Hash Scheduling memiliki response time rata-rata sebesar 2.96 ms, sedangkan algoritma URI menunjukkan response time rata-rata sebesar 3.1 ms. Seiring dengan peningkatan jumlah koneksi algoritma Source Hash Scheduling cenderung memberikan response time yang lebih rendah pada koneksi yang lebih tinggi, sementara algoritma URI menunjukkan kenaikan response time yang lebih stabil. Analisis ini memberikan wawasan penting mengenai seleksi algoritma load balancing untuk optimalisasi response time, yang krusial dalam meningkatkan kepuasan pengguna web server.

Kata kunci: *Load balancing, Response time, Source Hash Scheduling, URI, Web Server*

Abstract

The rapid growth of the internet has led to a significant increase in connected user access. This phenomenon directly affects the needs of service provider machines, especially web servers. This study aims to analyze the performance of web servers by implementing load-balancing strategies using two algorithms, they are Source Hash Scheduling and Uniform Resource Identifier (URI). Load balancing was tested to determine its effectiveness in optimizing the response time. Data collected from the tests showed that the average response time was in milliseconds for various connection levels, ranging from 100/100 to 5000/500. The test results indicated that at a connection level of 1000/100, the Source Hash Scheduling algorithm had an average response time of 2.96 ms, whereas the URI algorithm showed an average response time of 3.1 ms. As the number of connections increased the Source Hash Scheduling algorithm tending to provide lower response times at higher connections, whereas the URI algorithm showed a more stable increase in response time. This analysis provides valuable insights into the selection of load-balancing algorithms for optimizing response time, which is crucial for enhancing user satisfaction with web servers.

Keywords: *Load balancing, Response time, Source Hash Scheduling, URI, Web Server*

PENDAHULUAN

Seiring dengan perkembangan internet yang terus menerus dan cepat, terjadi peningkatan yang signifikan dalam akses pengguna ke layanan yang terhubung ke jaringan. Hal ini menghasilkan beragam layanan baru, seperti aplikasi web dalam berbagai bentuk, yang memicu peningkatan permintaan atas layanan web. Model-model layanan aplikasi web yang populer mencakup e-business, e-learning, e-news, di antara lainnya [1]. Perkembangan ini telah memicu kemunculan teknologi inovatif yang dikenal sebagai komputasi awan (*cloud computing*). Komputasi awan didefinisikan sebagai himpunan sumber daya komputasi, jaringan, solusi manajemen penyimpanan, dan dukungan untuk aplikasi yang berbasis virtual [2]. Tersedianya dapat disesuaikan dengan permintaan dan mempertimbangkan berbagai aspek, termasuk faktor ekonomi. Komputasi awan memiliki peran kunci dalam membangun infrastruktur TI yang dinamis, memastikan kualitas layanan, dan menyederhanakan konfigurasi layanan aplikasi [3]. Pengelolaan server memainkan peran krusial dalam menyediakan layanan website yang efisien, andal, dan *scalable* [4]. Dalam konteks komputasi awan, server tidak lagi bersifat fisik melainkan virtual, memungkinkan penyedia layanan untuk menyediakan sumber daya komputasi secara dinamis sesuai dengan kebutuhan. Peningkatan kinerja sistem layanan web server sangat tergantung pada kualitas sistem server itu sendiri. Tujuannya adalah untuk menanggulangi lonjakan permintaan yang terjadi pada server web selama periode akses yang tinggi. Sehingga, arsitektur yang terdiri dari beberapa server menjadi pilihan yang tepat melalui implementasi *load balancing*.

Dalam rangka meningkatkan performa server web yang sering menerima permintaan dalam jumlah besar, keberadaan sistem server yang stabil dan efisien menjadi esensial. *Load balancing* merupakan suatu teknik yang digunakan dalam sistem komputer untuk mendistribusikan beban kerja secara merata di antara beberapa sumber daya atau server guna mencegah terjadinya *overloading* pada satu atau beberapa server tertentu [5,6]. Penggunaan *load balance* pada jaringan sangat dibutuhkan jika jaringan tersebut aktif dan banyak diakses oleh pengguna, karena dapat menyebabkan ketidakseimbangan jaringan [7]. Teknologi *load balancing* didesain untuk alokasi beban koneksi yang adil dan merata di antara semua server yang beroperasi, sehingga memastikan distribusi yang seimbang [8,9,20]. *Load balancing*, sebagai teknik untuk meningkatkan ketersediaan dan kinerja server, melibatkan distribusi akses layanan secara serentak ke beberapa server, mencegah beban berlebih pada server tunggal [10,11,18]. Terdapat beberapa metode yang dapat diterapkan pada *load balancing*, yaitu metode *Source Hash Scheduling* [19] dan URI *Uniform Resource Identifier* (URI). Berdasarkan penelitian sebelumnya menunjukkan bahwa metode *Source Hash Scheduling* memiliki kemampuan dalam mengalokasikan beban kerja dengan melibatkan penggunaan informasi seperti alamat IP atau MAC *address* dari sumber permintaan [12]. Sedangkan URI memiliki kemampuan dalam menggunakan informasi pada level aplikasi, yaitu URI atau *path* dari permintaan [13]. URI yang dianalisis dari setiap permintaan membuat *load balancer* dapat menentukan server yang paling sesuai untuk menangani permintaan tersebut.

Namun, kedua metode ini perlu dikaji terkait hubungannya terhadap pengelolaan beban yang meningkat saat jumlah akses layanan naik pada kumpulan server. Kumpulan server ini, yang dikenal sebagai *cluster* server, diharapkan dapat meningkatkan ketersediaan aplikasi dan keandalan sistem secara keseluruhan [14]. Oleh karena itu diperlukan model sistem server yang tepat agar bisa menangani lonjakan permintaan tersebut. Perbandingan antara algoritma *Source Hash Scheduling* dan URI (*Uniform Resource Identifier*) dalam mendistribusikan beban yang diterima, perlu dilakukan untuk mengetahui efektivitas dari nilai *response time* yang didapat. Sehingga, penelitian ini bertujuan untuk merancang, menguji, dan mengevaluasi metode *load balancing* dengan membandingkan dua buah algoritma distribusi dalam meningkatkan kualitas layanan (QoS) layanan dalam menghadapi permintaan tinggi pada server web. Hasil penelitian

ini dapat menjadi informasi metode *load balancing* yang sesuai untuk parameter *response time* yang akan diamati.

METODE PENELITIAN

Load balancing dalam penelitian ini diterapkan untuk menilai perbaikan performa *server web* dengan membandingkan kondisi operasional sebelum dan setelah *load balancing* diterapkan. Tahap permulaan penelitian ini mencakup review literatur untuk membangun pemahaman mendalam tentang subjek. Tahapan berikutnya, penelitian melanjutkan dengan pengaturan web server dan penerapan teknik *load balancing*. Kinerja *web server* kemudian diuji coba dengan serangkaian beban koneksi, yang diterapkan pada konfigurasi server tunggal dan juga pada konfigurasi multi-server.

Evaluasi diarahkan pada efektivitas metode *load balancing*, menggunakan kriteria dan indikator yang telah ditentukan sebelumnya. Hasil dari proses evaluasi ini akan menampilkan kinerja sesuai dengan indikator yang telah diukur. Seluruh prosedur penelitian ini digambarkan dalam sebuah diagram alir pada Gambar 1, yang memvisualisasikan alur metodologis yang diikuti dalam penelitian ini.

Gambar 1 Metode Penelitian

Studi Literatur

Di fase tahapan penelitian ini ialah cara mengumpulkan data dari berbagai sumber tertulis, yang meliputi buku, jurnal ilmiah, artikel, dan dokumen-dokumen relevan lainnya [15].

Konfigurasi *Server web*

Tahapan ini terfokus pada pengaturan dan konfigurasi server web. Pada penelitian ini, *server web* yang digunakan adalah Apache, yang dikenal karena kinerjanya yang baik. Selain itu, *server load balancing* yang digunakan adalah HAproxy, yang berfungsi untuk mendistribusikan beban kerja secara merata di antara beberapa *server backend.*

Konfigurasi *Server Load balancing*

Langkah ini melibatkan pengaturan server *load balancing* yang bertujuan untuk meningkatkan efisiensi dalam distribusi beban kerja. Server *load balancing* ini dibangun menggunakan HAproxy, sebuah *software* yang terkenal dalam mengelola lalu lintas jaringan dan memastikan bahwa permintaan klien didistribusikan secara merata ke *server backend*.

Pada implementasi ini menggunakan dua mekanisme algoritma, yaitu *Source Hash Scheduling* dan URI *Hash Scheduling*. Algoritma *Source Hash Scheduling* bekerja dengan cara mendistribusikan permintaan berdasarkan hash dari alamat IP sumber, sehingga memastikan bahwa permintaan dari sumber yang sama akan diarahkan ke server yang sama. Ini sangat berguna untuk aplikasi yang membutuhkan sesi yang konsisten. Sementara itu, algoritma URI *Hash Scheduling* mendistribusikan permintaan berdasarkan *hash* dari *Uniform Resource Identifier* (URI) permintaan. Dengan cara ini, permintaan untuk konten yang sama akan selalu diarahkan ke server yang sama, yang berguna untuk *caching* dan efisiensi data. Kedua algoritma yang digunakan ini diharapkan beban kerja dapat didistribusikan dengan lebih efektif, mengurangi waktu respon, dan meningkatkan kinerja server secara keseluruhan.

Pengujian dan Evaluasi Kinerja Server

Fase ini menguji server untuk menilai performanya sebelum dan setelah implementasi *load balancing.* Dalam studi ini, permintaan dilakukan dengan tingkat koneksi yang berbeda, mulai dari 1000/100 hingga 5000/500 koneksi, dengan tujuan untuk mengukur kinerja dari *response time*. Hasil pengujian kemudian dianalisis untuk mendapatkan kesimpulan yang komprehensif.

Dokumentasi Hasil Pengujian

Fase ini mencakup pencatatan dan dokumentasi dari hasil pengujian, yang menjadi sumber data berharga untuk rujukan di masa depan serta analisis mendalam yang akan datang.

Spesifikasi Sistem Server

Studi ini menerapkan *Virtual Machine* (VM) sebagai pondasi dari sistem server yang dibangun. Penelitian ini melibatkan tiga perangkat server yang dibangun secara *virtual*. Spesifikasi untuk setiap server termasuk memori RAM sejumlah 1 GB, prosesor CPU dengan 1 inti, kapasitas penyimpanan SSD sebanyak 20 GB. Perangkat utama yang digunakan dalam studi ini adalah AMD Ryzen 5 2500U with Radeon Vega Mobile Gfx dengan memori RAM 8 GB.

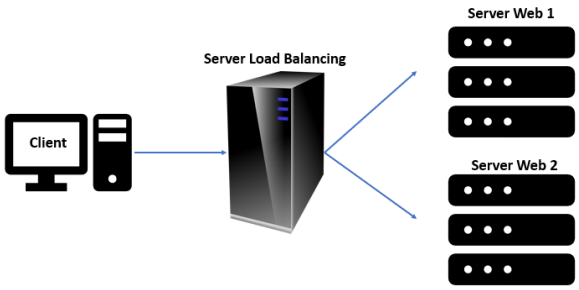
HASIL DAN PEMBAHASAN

Teknik *load balancing* merupakan metode yang digunakan dalam jaringan komputer untuk mendistribusikan beban permintaan yang diterima ke berbagai komputer atau kelompok komputer yang bekerja sebagai satu kesatuan [16]. Kegunaan utama dari teknik ini adalah untuk memaksimalkan efisiensi penggunaan sumber daya, meningkatkan jumlah pemrosesan data (*throughput*), meminimalkan waktu tanggapan, dan mencegah terjadinya kelebihan beban pada sistem [6]. Sebuah *cluster* server adalah sekumpulan komputer yang terkoneksi satu sama lain dan bekerja secara sinergis dalam berbagai fungsi, menciptakan sebuah sistem yang terkoordinasi. *Cluster* komputer ini biasanya dirancang untuk memastikan ketersediaan tinggi dan performa yang lebih baik dari sistem komputer [1].

Fungsi lain dari *load balancing* meliputi pemantauan kondisi server, baik itu dari aspek aplikasi maupun konten, yang berperan dalam meningkatkan ketersediaan layanan dan kemudahan dalam manajemen sistem [17]. Dalam konteks yang lebih luas, *load balancing* tidak hanya bertindak sebagai penyeimbang beban, tapi juga sebagai pengelola arus data dan pemandu alih jalur trafik jaringan.

Tujuan dari pengujian parameter pada sistem *load balancing* adalah untuk mengukur efisiensi server dalam menangani berbagai permintaan dari pengguna dalam jangka waktu yang ditentukan. Pengujian ini fokus pada pengukuran *response time* menggunakan dua algoritma distribusi beban yang berbeda. Studi ini memanfaatkan VM untuk membangun dan meniru lingkungan infrastruktur server. Arsitektur dari sistem server ini digambarkan dalam Gambar 2, yang menggambarkan implementasi server *load balancing* bersama dengan dua web server, memungkinkan permintaan dari pengguna untuk diproses tidak hanya oleh satu server, namun disebar ke dua atau lebih server untuk meningkatkan distribusi dan efisiensi layanan.

Saat layanan pada web server diakses oleh pengguna, server *load balancer* menjadi perantara pertama yang mengolah permintaan tersebut sebelum mengirimkannya ke web server yang sesuai. Dalam proses ini, informasi web server yang menerima permintaan pengguna tidak diketahui oleh pengguna itu sendiri. Server *load balancer* bertugas mendistribusikan permintaan tersebut berdasarkan algoritma distribusi yang digunakan ke semua web server yang tercatat dalam sistem distribusinya. Alamat IP dari antarmuka jaringan yang terpasang diinformasikan pada Tabel 1.



Gambar 2 Arsitektur server dengan *load balancing*

Tabel 1 Nama Perangkat dan *IP Address*

Nama Perangkat	*IP Address*
Load Balancer	192.168.109.184
Server Web 1	192.168.109.179
Server Web 2	192.168.109.178
Pengguna	192.168.109.182

Pada Tabel 1 menerangkan bahwa server *load balancer*, yang beralamat IP 192.168.109.184, berperan dalam mengalokasikan permintaan pengguna ke berbagai web server. Penelitian ini melaksanakan evaluasi kinerja web server dengan mengukur *response time* setelah implementasi algoritma *Source Hash Scheduling* dan URI untuk *load balancing*. Pengujian menggunakan alat httperf yang dijalankan oleh pengguna untuk mengirim permintaan simultan, mencatat nilai *response time*. Hasil pengujian, yang menampilkan *response time* saat mengirim permintaan ke web server, tersaji dalam Tabel 2. Permintaan diuji secara bertingkat, mulai dari 1000 permintaan per 100 koneksi per detik hingga 5000 permintaan per 500 koneksi per detik, memungkinkan analisis perbandingan kinerja antara kedua algoritma berdasarkan data *response time* yang terkumpul.

Tabel 2 Hasil pengujian terhadap algoritma *Source Hash Scheduling* dan URI

No	Tingkat Koneksi	*Source Hash Scheduling* (ms)					URI (ms)				
		Pengujian					Pengujian				
		ke-1	ke-2	ke-3	ke-4	ke-5	ke-1	ke-2	ke-3	ke-4	ke-5
1	1000/100	2.9	3.1	3.1	2.9	2.8	3.2	3	3.6	2.9	2.8
2	2000/200	3	3.1	3.6	2.8	3	2.9	2.9	2.9	2.9	3.3
3	3000/300	2.7	2.8	2.8	2.7	2.7	2.8	2.7	4.6	3.5	2.9
4	4000/400	4	3.5	3.6	3.5	3.6	3.6	3.8	3.6	3.5	3.8
5	5000/500	3.5	3.5	3.5	3.5	3.5	3.5	3.8	3.8	3.8	3.8

Tabel 2 menampilkan hasil pengujian *response time* dari sebuah web server yang mengimplementasikan *load balancing* dengan dua algoritma berbeda, yaitu algoritma *Source Hash Scheduling* dan algoritma URI. Hasil rata-rata pengujian ditunjukkan dalam milidetik *(ms)* dan diurutkan berdasarkan tingkat koneksi yang berbeda, dari 1000/100 hingga 5000/500.

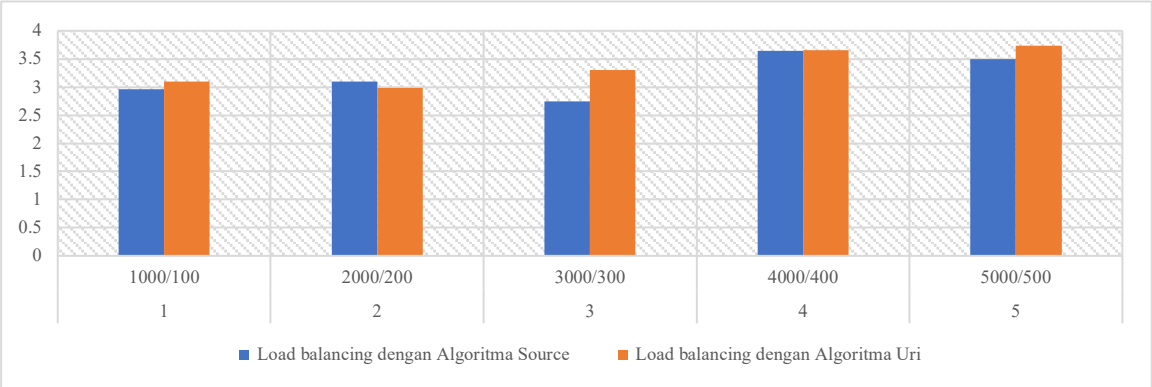
Dari data yang ada, dapat dilihat bahwa untuk tingkat koneksi yang lebih rendah (1000/100 dan 2000/200), *response time* yang dihasilkan oleh kedua algoritma cenderung serupa, dengan algoritma *Source Hash Scheduling* memberikan performa yang sedikit lebih baik pada tingkat koneksi 1000/100. Namun, seiring dengan peningkatan jumlah koneksi dimulai dari 3000/300, algoritma *Source Hash Scheduling* cenderung memiliki *response time* yang lebih rendah dibandingkan dengan algoritma URI, menunjukkan bahwa algoritma *Source Hash Scheduling* mungkin lebih efisien dalam menangani beban yang lebih tinggi. Peningkatan jumlah koneksi juga berdampak pada peningkatan *response time* secara umum untuk kedua algoritma, tetapi peningkatan ini lebih curam pada algoritma URI dibandingkan dengan algoritma *Source Hash Scheduling*. Ini mengindikasikan bahwa algoritma *Source Hash Scheduling* lebih *scalable* dan dapat menjaga kinerja yang lebih konsisten pada beban yang lebih tinggi.

Berdasarkan Tabel 3 kemudian dibuat dalam bentuk grafik yang menggambarkan perbandingan dengan mengacu pada hasil pengujian yang tersaji pada Gambar 3. Gambar 3 menunjukkan grafik yang membandingkan kualitas layanan (QoS) dari dua algoritma *load balancing* berdasarkan *response time* dalam milidetik. Ada lima pasang batang yang mencerminkan lima level koneksi berbeda, mulai dari 1000/100 hingga 5000/500.

Tabel 3 Hasil Rata-rata Pengujian *Response time* (*ms*)

No	Tingkat Koneksi	Hasil Rata-rata Pengujian *Response time* (*ms*)	
		Source Hash Scheduling	**URI**
1	1000/100	2.96	3.1
2	2000/200	3.1	2.98
3	3000/300	2.75	3.3
4	4000/400	3.64	3.66
5	5000/500	3.5	3.74

Pada koneksi rendah, kedua algoritma menunjukkan *response time* yang hampir sama, namun pada koneksi yang lebih tinggi, algoritma *Source Hash Scheduling* menunjukkan *response time* yang konsisten lebih rendah, menandakan performa QoS yang lebih baik dalam hal waktu respon. Grafik ini memberikan visualisasi langsung dari efektivitas kedua algoritma dalam menangani berbagai beban koneksi dalam konteks QoS.



Gambar 3 Grafik dari *Response time (ms)*

Secara keseluruhan, analisis ini menunjukkan bahwa implementasi *load balancing* dengan algoritma *Source Hash Scheduling* bisa menjadi pilihan yang lebih baik untuk optimalisasi *response time*, khususnya saat web server menghadapi jumlah koneksi yang tinggi. Namun, kedua algoritma tersebut menunjukkan keefektifan yang kompetitif pada tingkat koneksi yang lebih rendah. Dalam praktiknya, pilihan algoritma mungkin juga akan dipengaruhi oleh faktor-faktor lain seperti jenis *traffic*, konfigurasi server, dan sumber daya yang tersedia.

KESIMPULAN

Hasil pengujian *response time* pada web server dengan algoritma *Source Hash Scheduling* dan URI menunjukkan variasi performa tergantung tingkat koneksi. Pada koneksi rendah 1000/100 dan 2000/200, kedua algoritma hampir serupa, namun *Source Hash Scheduling* unggul sedikit pada 1000/100. Peningkatan dari 3000/300 dan seterusnya, *Source Hash Scheduling* secara konsisten menunjukkan *response time* lebih rendah dibandingkan URI, mengindikasikan efisiensi lebih tinggi. Kenaikan jumlah koneksi meningkatkan *response time* pada kedua algoritma, tetapi kenaikan pada URI lebih tajam. Kenaikan ini menandakan bahwa *Source Hash Scheduling* lebih *scalable* dan stabil di bawah beban tinggi. Oleh karena itu, *Source Hash Scheduling* lebih menguntungkan untuk optimalisasi *response time*, terutama pada trafik web padat. Namun, pemilihan algoritma juga harus mempertimbangkan faktor lain seperti jenis *traffic*, konfigurasi server, dan sumber daya yang tersedia.

DAFTAR PUSTAKA

1. S. D. Riskiono and D. Pasha, “Analisis Metode Load Balancing Dalam Meningkatkan Kinerja Website E-Learning,” *J. Teknoinfo*, vol. 14, no. 1, pp. 22–26, 2020, doi: 10.33365/jti.v14i1.466.

2. S. Afzal and G. Kavitha, "Load balancing in cloud computing – A hierarchical taxonomical classification," *J. Cloud Comput.*, vol. 8, no. 1, 2019, doi: 10.1186/s13677-019-0146-7.
3. Y. Afrianto and A. H. Hendrawan, "Implementasi Data Center Untuk Penempatan Host Server Berbasis Private Cloud Computing," *Krea-Tif*, vol. 7, no. 1, p. 50, 2019, doi: 10.32832/kreatif.v7i1.2031.
4. I. Ahmad, E. Suwarni, R. I. Borman, A. Asmawati, F. Rossi, and Y. Jusman, "Implementation of RESTful API Web Services Architecture in Takeaway Application Development," in *International Conference on Electronic and Electrical Engineering and Intelligent System (ICE3IS)*, 2022, pp. 132–137.
5. M. Jamil, S. Santosa, and M. Hamid, "Analisis Perbandingan Kinerja Load Balancing Menggunakan Metode PCC dan NTH," *J. PRODUKTIF*, vol. 7, no. 1, pp. 619–625, 2023.
6. W. Wartono, M. H. Prayitno, and J. Trianto, "Analisis Performa Load Balancing Terhdap Throughput Pada Kluster Server Untuk Mendukung Smart City," *J. Teknoinfo*, vol. 18, no. 1, pp. 173–181, 2024.
7. T. Hidayat, Y. Azzery, and R. Mahardiko, "Load Balancing Network by using Round Robin Algorithm: A Systematic Literature Review," *J. Online Inform.*, vol. 4, no. 2, p. 85, 2020, doi: 10.15575/join.v4i2.446.
8. T. Octavriana, K. Joni, and A. F. Ibadillah, "Optimalisasi Jaringan Internet Dengan Load Balancing Pada High Traffic Network," *J. Tek. Inform.*, vol. 14, no. 1, pp. 28–39, 2021, doi: 10.15408/jti.v14i1.15018.
9. M. A. I. F. P. Sujarwo, I. Istikmal, and A. I. Irawan, "Analysis of Load Balancing Least Connection and Shortest Expected Delay Algorithm for Web Server Using Kube-Proxy on Kubernetes," *ELKOMIKA J. Tek. Energi Elektr. Tek. Telekomun. Tek. Elektron.*, vol. 11, no. 2, p. 439, 2023, doi: 10.26760/elkomika.v11i2.439.
10. H. S. Harefa, J. Triyono, and S. Raharjo, "Implementasi Load Balancing Web Server Untuk Optimalisasi Kinerja Web Server Dan Database," *J. Jarkom*, vol. 09, no. 01, pp. 10–20, 2021.
11. 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.
12. A. Sumiati, P. Hari Trisnawan, and M. Ali Fauzi, "Implementasi Load Balancing Web Server dengan Algoritma Source IP Hash pada Software Defined Network (SDN)," *J. Pengemb. Teknol. Inf. dan Ilmu Komput.*, vol. 4, no. 3, pp. 919–928, 2020.
13. A. M. Pradana, T. W. Purboyo, and R. Latuconsina, "Analisis Load Balancing Pada Jaringan Software Defined Network (SDN) Menggunakan Algoritma Jaringan Syarag Tiruan (JST)," *e-Proceeding Eng.*, vol. 6, no. 1, pp. 1393–1400, 2019.
14. R. Nuraini, "Implementasi Metode Load Balancing Untuk Peningkatan Nilai Troughput Pada Server," *Kumpul. J. Ilmu Komput.*, vol. 09, no. 03, pp. 467–478, 2022.
15. A. Amarudin and S. D. Riskiono, "Analisis Dan Desain Jalur Transmisi Jaringan Alternatif Menggunakan Virtual Private Network (Vpn)," *J. Teknoinfo*, vol. 13, no. 2, p. 100, 2019, doi: 10.33365/jti.v13i2.309.
16. T. Wira Harjanti, H. Setiyani, and J. Trianto, "Load Balancing Analysis Using Round-Robin and Least-Connection Algorithms for Server Service Response Time," *Appl. Technol. Comput. Sci. J.*, vol. 5, no. 2, pp. 40–49, 2022, doi: 10.33086/atcsj.v5i2.3743.
17. B. G. Malau, "Implementasi Load Balancing Mikrotik Jaringan Internet Di Pardamean Sibisa, Ajibata, Toba Samosir, Sumatra Utara," *JCS-TECH J. Comput. Sci. Technol.*, vol. 2, no. 1, pp. 20–29, 2022, doi: 10.54840/jcstech.v2i1.23.

18. Riskiono, S. D., & Darwis, D. (2020). Peran Load Balancing Dalam Meningkatkan Kinerja Web Server Di Lingkungan Cloud. Krea-TIF: Jurnal Teknik Informatika, 8(2), 1–8. https://doi.org/10.32832/kreatif.v8i2.3503
19. R. A. Setyawan, "Analisis Implementasi Load Balancing dengan Metode Source Hash Scheduling pada Procol SSL", jeeccis, vol. 8, no. 2, pp. pp.204–208, Aug. 2014.
20. Z. Saharuna, R. Nur, and A. Sandi, "Analisis Quality Of Service Jaringan Load Balancing menggunakan Metode PCC dan NTH," CESS (Journal Comput. Eng. Syst. Sci., vol. 5, no.1, p. 131, 2020, doi: 10.24114/cess.v5i1.14629.