JURNAL ILMIAH FIFO **P-ISSN 2085-4315 / E-ISSN 2502-8332**
DOI: http://dx.doi.org/10.22441/fifo.2024.v16i2.008

Analisis Perbandingan Algoritma *Load Balancing Source Hash Scheduling* dan URI Berdasarkan *Throughput* Pada Server Web

Hendra Mayatopani[1]*, Arief Herdiansah[2], Sofyan[3], Faiz Muqorrir Kaaffah[4]

Program Studi Sistem Informasi, Fakultas Sains dan Teknologi, Universitas Pradita[1]
Program Studi Teknik Informatika, Fakultas Teknik, Universitas Muhammadiyah Tangerang[2]
Program Studi Ilmu Komputer, STMIK Kreatindo Manokwari[3]
Program Studi Informatika, UIN Siber Syekh Nurjati Cirebon[4]
hendra.mayatopani@pradita.ac.id[1*], arief_herdiansah@umt.ac.id[2], sofyanarifin018@gmail.com[3], faiz@syekhnurjati.ac.id[4]

*) Corresponding Author

(***received*:** 15-08-24, ***revised*:** 23-08-24, ***accepted*:** 07-09-24)

Abstract

Along with the rapid development of the internet, there has been a significant increase in the number of connected users, which directly impacts the performance needs of web servers. This research aims to evaluate web server performance through the implementation of Load Balancing methods using two different algorithms: Source Hash Scheduling (SHS) and Uniform Resource Identifier (URI). The Load Balancing method was chosen for its ability to distribute workload evenly across multiple servers, thereby enhancing system efficiency and availability. The SHS algorithm was selected for its capability to ensure consistent distribution of requests based on source IP addresses, while the URI algorithm was chosen for its ability to distribute requests based on more specific URI patterns. Experiments were conducted to measure the effectiveness of both algorithms in improving throughput at various connection levels. The analysis results show that both algorithms deliver excellent and consistent performance. At low connections (1000/100), they recorded identical throughput of 51.20 KB/s. However, at higher connection levels (2000/200 to 5000/500), URI slightly outperformed SHS with throughput up to 255.70 KB/s, compared to 255.74 KB/s for SHS. Although the performance difference is minimal, URI demonstrated better stability in maintaining throughput. Therefore, the choice of algorithm can be adjusted to specific needs related to stability or performance at certain connection levels, with both algorithms offering reliable solutions to enhance web server performance and user satisfaction.

***Keywords**: Load Balancing, Source Hash Scheduling, URI (Uniform Resource Identifier), Throughput, Server Web Performance*

Abstrak

Seiring dengan pesatnya perkembangan internet, terjadi peningkatan signifikan dalam jumlah pengguna yang terhubung, yang berdampak langsung pada kebutuhan performa server web. Penelitian ini bertujuan untuk mengevaluasi kinerja server web melalui penerapan metode *Load Balancing* menggunakan dua algoritma berbeda, yaitu *Source Hash Scheduling* (SHS) dan *Uniform Resource Identifier* (URI). Metode *Load Balancing* dipilih karena kemampuannya dalam mendistribusikan beban kerja secara merata di antara beberapa server, sehingga dapat meningkatkan efisiensi dan ketersediaan sistem. Algoritma SHS digunakan karena kemampuannya dalam memastikan konsistensi distribusi permintaan berdasarkan alamat IP sumber, sementara algoritma URI dipilih karena dapat mendistribusikan permintaan berdasarkan pola URI yang lebih spesifik. Uji coba dilakukan untuk mengukur efektivitas kedua algoritma dalam meningkatkan *throughput* pada berbagai tingkat koneksi. Hasil analisis menunjukkan bahwa kedua algoritma memberikan performa yang sangat baik dan konsisten. Pada koneksi rendah (1000/100), keduanya mencatat throughput identik sebesar 51.20 KB/s. Namun, pada tingkat koneksi lebih tinggi (2000/200 hingga 5000/500), URI sedikit lebih unggul dengan throughput hingga 255.70 KB/s, dibandingkan dengan 255.74 KB/s pada SHS. Meskipun perbedaan performa sangat kecil, URI menunjukkan stabilitas yang lebih baik dalam menjaga *throughput*. Oleh karena itu, pemilihan algoritma dapat disesuaikan dengan kebutuhan spesifik terkait stabilitas atau performa pada tingkat koneksi tertentu, dengan kedua algoritma ini menawarkan solusi andal untuk meningkatkan kinerja dan kepuasan pengguna server web.

Kata Kunci: *Load Balancing*, *Source Hash Scheduling*, URI (*Uniform Resource Identifier*), *Throughput*, Server Web Performance

I. Pendahuluan

Ekspansi internet yang cepat telah memicu peningkatan dramatis dalam jumlah pengguna yang terhubung ke jaringan global. Lonjakan jumlah pengguna ini tidak hanya mendorong munculnya berbagai layanan baru tetapi juga meningkatkan permintaan terhadap layanan web yang sudah ada. Model aplikasi web yang populer seperti e-bisnis, e-learning, dan e-berita kini semakin sering diadopsi dan menjadi bagian integral dari kehidupan sehari-hari. Adopsi luas ini mencerminkan bagaimana teknologi digital telah meresap ke dalam berbagai aspek kehidupan dan bisnis [1]. Perkembangan pesat ini juga telah menyebabkan munculnya teknologi inovatif yang revolusioner, dikenal sebagai komputasi awan. Komputasi awan, yang menawarkan kumpulan sumber daya komputasi yang luas, infrastruktur jaringan, solusi manajemen penyimpanan, dan dukungan yang kuat untuk aplikasi dalam lingkungan yang tervirtualisasi, telah menjadi katalis dalam transformasi teknologi informasi. Dengan komputasi awan, sumber daya ini dapat disesuaikan secara fleksibel untuk memenuhi kebutuhan pengguna, dengan mempertimbangkan berbagai faktor, termasuk pertimbangan ekonomi. Ini memungkinkan perusahaan dan individu untuk mengakses teknologi terbaru tanpa investasi besar dalam hardware fisik [2]. Signifikansi komputasi awan dalam membangun infrastruktur TI yang fleksibel tidak bisa diremehkan. Ini tidak hanya memastikan kualitas layanan yang konsisten tetapi juga merampingkan konfigurasi dan pengelolaan aplikasi, yang merupakan aspek penting dalam manajemen IT yang efisien. Dalam konteks ini, manajemen server yang efektif menjadi sangat penting, karena layanan web yang efisien, dapat diandalkan, dan skalabel sangat bergantung pada infrastruktur yang mendasarinya [3].

Di era komputasi awan ini, server telah mengalami transformasi dari perangkat keras fisik menjadi entitas yang sepenuhnya tervirtualisasi. Ini memungkinkan penyedia layanan untuk mengalokasikan dan mengelola sumber daya komputasi secara dinamis berdasarkan permintaan yang berfluktuasi. Hal ini sangat kritikal saat menghadapi lonjakan permintaan, terutama selama waktu penggunaan puncak, yang dapat berdampak signifikan pada performa sistem layanan web [4]. Oleh karena itu, mengadopsi arsitektur dengan penerapan multiple server dan *Load Balancing* menjadi strategi yang optimal. Strategi ini tidak hanya meningkatkan kapasitas dan keandalan layanan tetapi juga memastikan bahwa sumber daya dapat ditambah atau dikurangi secara cepat dan efisien sesuai dengan kebutuhan, sehingga memaksimalkan efisiensi operasional dan mengurangi potensi gangguan layanan. Dengan cara ini, organisasi dapat menjamin layanan yang responsif dan berkelanjutan untuk pengguna mereka, terlepas dari fluktuasi dalam permintaan atau beban kerja.

Meningkatkan kinerja server web yang sering mengalami lonjakan permintaan membutuhkan infrastruktur server yang dapat diandalkan dan efisien. *Load Balancing* adalah teknik penting yang mendistribusikan beban kerja ke beberapa server atau sumber daya, yang secara efektif mencegah server tunggal menjadi kelebihan beban [5], [6]. Teknik ini sangat penting, terutama di jaringan dengan aktivitas pengguna yang tinggi dan akses yang sering, karena tidak adanya *Load Balancing* dapat menyebabkan ketidakstabilan dan ketidakseimbangan jaringan [7]. Teknologi *Load Balancing* menjamin distribusi beban koneksi yang merata di semua server yang aktif, sehingga menjaga sistem yang seimbang dan stabil [8], [9]. *Load Balancing* memungkinkan permintaan layanan tersebar secara merata di beberapa server, secara efektif mencegah satu server menjadi kewalahan [10], [11]. Berbagai teknik *Load Balancing*, termasuk *Source Hash Scheduling* (SHS) dan *Uniform Resource Identifier* (URI), dapat diimplementasikan. Penelitian sebelumnya menunjukkan bahwa Penjadwalan *Source Hash Scheduling* (SHS) dapat mengalokasikan beban kerja menggunakan detail seperti alamat IP atau alamat MAC asal permintaan [12]. Di sisi lain, URI memanfaatkan detail tingkat aplikasi, seperti jalur permintaan atau URI, untuk mengidentifikasi server yang paling sesuai untuk memproses permintaan [13].

Meskipun demikian, analisis mendalam terhadap kedua metode *Load Balancing*, SHS dan URI, menjadi sangat penting dalam konteks peningkatan beban yang dihadapi ketika jumlah akses layanan pada kluster server mengalami peningkatan yang signifikan. Kluster server ini dirancang khusus untuk meningkatkan aksesibilitas aplikasi dan memperkuat keandalan sistem secara keseluruhan [14], sehingga memastikan bahwa sistem dapat beroperasi secara efisien bahkan di bawah kondisi trafik yang sangat tinggi.

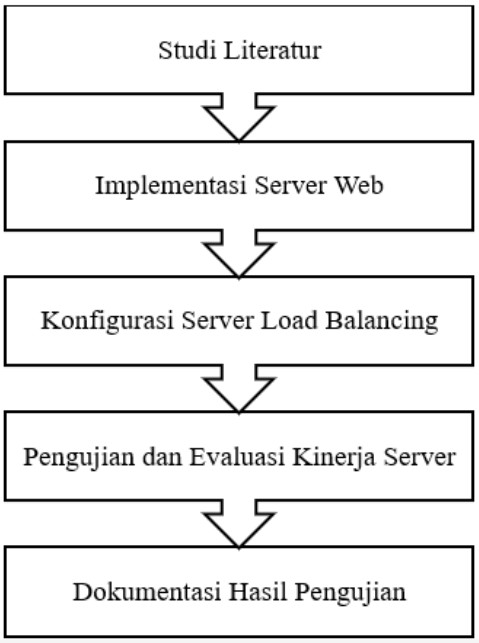
Dalam situasi di mana lonjakan permintaan menjadi rutinitas, memiliki model sistem server yang tangguh dan mampu menangani beban tersebut dengan efektif adalah krusial. Evaluasi kinerja dan efektivitas algoritma *Load Balancing* SHS dan URI dalam mengatur dan mendistribusikan beban ini menjadi sangat vital [15]. Analisis ini

tidak hanya melihat pada nilai *throughput* yang dihasilkan, tetapi juga pada seberapa baik setiap algoritma dapat menjaga atau bahkan meningkatkan kualitas layanan (*Quality of Service*/QoS) selama periode trafik puncak.

Oleh karena itu, penelitian ini bertujuan untuk merancang, menguji, dan menilai teknik *Load Balancing* dengan membandingkan kinerja kedua algoritma tersebut. Fokus utama dari penelitian ini adalah pada peningkatan kualitas layanan untuk server web, yang diukur melalui pengukuran *throughput* serta parameter lain seperti waktu respons dan stabilitas koneksi selama kondisi beban tinggi. Melalui pendekatan ini, diharapkan dapat diperoleh insight yang lebih dalam mengenai cara-cara efektif untuk mengoptimalkan distribusi beban dalam kluster server, sehingga memungkinkan sistem untuk menangani permintaan yang meningkat tanpa mengorbankan kinerja atau keandalan.

II. Metodologi Penelitian

Sebagai langkah awal dalam memahami dan mengukur performa server web dengan berbagai konfigurasi *Load Balancing*, penelitian ini dilakukan melalui serangkaian tahapan yang sistematis. Bagian ini menjelaskan metodologi yang digunakan dalam penelitian, mulai dari studi literatur hingga implementasi dan pengujian server. Setiap langkah yang diambil telah dirancang untuk memastikan validitas dan reliabilitas hasil yang diperoleh, sehingga dapat memberikan wawasan yang jelas mengenai efektivitas algoritma *Load Balancing* yang diuji. Berikut adalah penjelasan rinci mengenai metodologi yang digunakan dalam penelitian ini, yang dijelaskan secara bertahap melalui diagram alur yang ditunjukkan pada Gambar 1.



Gambar 1. Tahapan penelitian

Dalam artikel jurnal ini, kami menguraikan metodologi penelitian yang digunakan untuk mempelajari efektivitas berbagai algoritma *Load Balancing* dalam pengelolaan server web. Metodologi ini dirancang untuk memberikan analisis komprehensif dan sistematis mengenai kinerja server dalam berbagai konfigurasi *Load Balancing*. Berikut adalah deskripsi terperinci dari setiap tahapan metodologi seperti yang terillustrasi dalam Gambar 1, yang menjadi acuan utama dalam penelitian ini:

1. Studi Literatur

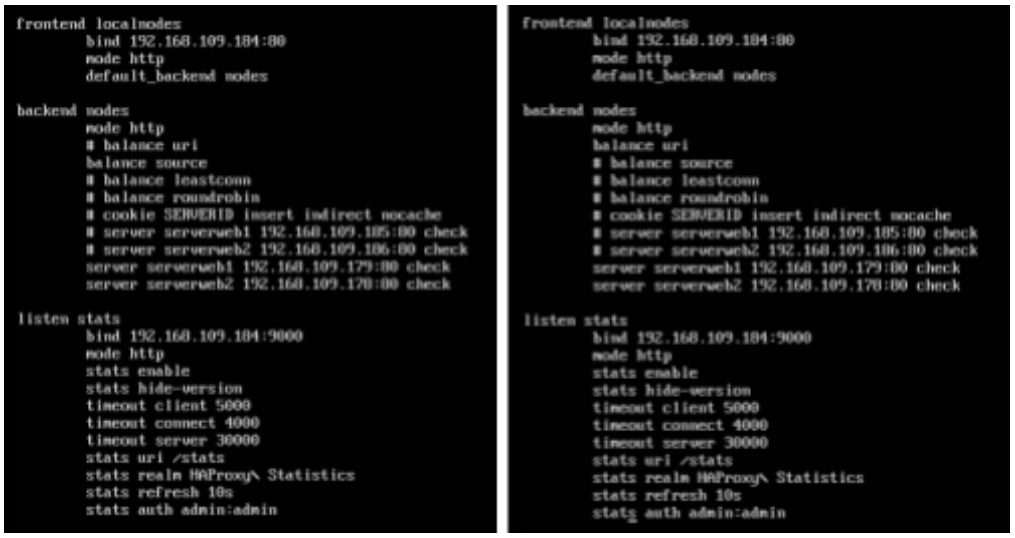
Tahapan ini merupakan langkah awal yang krusial, di mana tim peneliti melakukan kajian mendalam terhadap literatur yang ada. Tujuannya adalah untuk mengumpulkan informasi terkini dan relevan tentang teknologi *Load Balancing*, termasuk studi-studi terdahulu, teori-teori yang berkaitan, serta aplikasi teknologi tersebut dalam lingkungan industri. Literatur yang ditinjau meliputi jurnal-jurnal *peer-reviewed*, konferensi ilmiah, serta publikasi industri yang memiliki reputasi baik. Analisis literatur ini membantu dalam mendefinisikan masalah penelitian, mengidentifikasi gap dalam pengetahuan yang ada, serta merumuskan hipotesis dan pertanyaan penelitian yang spesifik.

2. Implementasi Server Web

Setelah pengumpulan dan analisis literatur, tahap berikutnya adalah implementasi fisik dari server web. Ini melibatkan konfigurasi hardware dan software yang diperlukan untuk mendukung lingkungan tes *Load Balancing*. Proses ini termasuk *setup* server, instalasi sistem operasi, dan konfigurasi jaringan yang sesuai. Kesiapan infrastruktur ini esensial untuk memastikan bahwa uji coba yang dilakukan dapat mencerminkan kondisi operasional yang realistis dan valid.

3. Konfigurasi Server *Load Balancing*

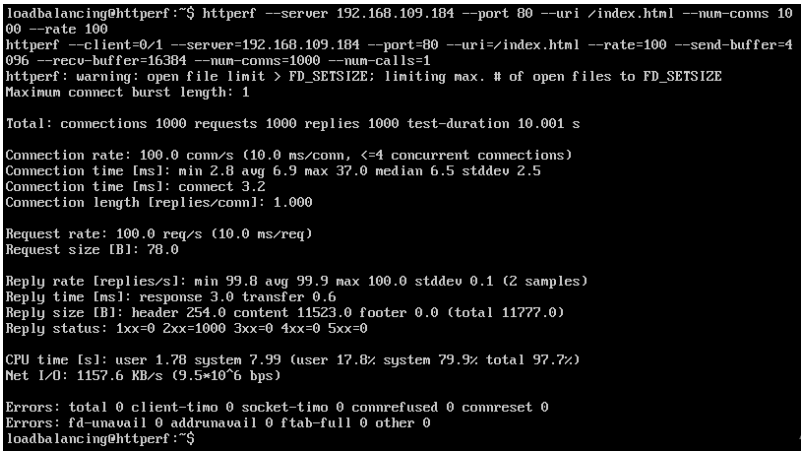
Tahap ini fokus pada konfigurasi spesifik dari algoritma *Load Balancing* yang akan diuji. Berbagai parameter teknis disesuaikan untuk mencerminkan berbagai skenario penggunaan yang akan dihadapi oleh server dalam produksi. Pemilihan algoritma berbasis prioritas penelitian dan relevansi terhadap kebutuhan industri saat ini, serta potensi inovasi dalam mengatasi kelemahan yang ada. Untuk konfigurasi *load balancing* untuk algoritma *source hash scheduling* dan *uniform resource identifier* ditunjukkan pada Gambar 2.



Gambar 2. Konfigurasi *Load Balancing*

4. Pengujian dan Evaluasi Kinerja Server

Setelah server dan algoritma *Load Balancing* dikonfigurasi, langkah selanjutnya adalah pengujian kinerja. Ini melibatkan simulasi berbagai beban kerja dan kondisi jaringan untuk mengevaluasi bagaimana algoritma *Load Balancing* mempengaruhi performa server. Metrik yang diukur meliputi *throughput* untuk melihat kehandalan algoritma terhadap beban yang meningkat. Pengujian ini menggunakan httperf untuk melakukan pembangkitan sejumlah *user*. Untuk melakukan pengujian dalam membangkitkan *request* dari *user* ditunjukkan pada Gambar 2.



Gambar 3. Pengujian Koneksi Menggunakan httperf

Dari Gambar 3 diatas, perintah `httperf` ini menguji kinerja server dengan alamat IP atau hostname `192.168.109.184` pada port `80`, yang merupakan port default untuk HTTP. Pengujian dilakukan dengan membuat total `1000` koneksi, di mana `100` koneksi per detik akan dibuat selama pengujian berlangsung.

5. Dokumentasi Hasil Pengujian

Semua data dan temuan dari pengujian kemudian didokumentasikan secara menyeluruh. Proses dokumentasi ini meliputi penyusunan data, analisis statistik, dan penulisan hasil dalam format yang sesuai untuk diseminasi ilmiah. Hasil ini kemudian ditinjau kritis untuk mengidentifikasi tren, menarik kesimpulan, dan merumuskan rekomendasi untuk penggunaan praktis teknologi *Load Balancing* dalam manajemen server.

Artikel ini menyajikan setiap tahapan dengan mendetail sesuai dengan prosedur ilmiah, menjamin bahwa metodologi yang digunakan dalam penelitian ini dapat diverifikasi dan diulang oleh peneliti lain, serta mendukung integritas dan keandalan kesimpulan yang ditarik dari studi ini. Gambar 1 dalam artikel ini bukan hanya sebagai elemen visual tetapi sebagai bagian penting dari eksposisi metodologis, memperjelas alur kerja dan menjamin transparansi proses penelitian.

Spesifikasi Sistem Server

Dalam penelitian ini, infrastruktur server diimplementasikan menggunakan *Virtual Machine* (VM) sebagai platform utama, memberikan fleksibilitas dan efisiensi dalam pengelolaan sumber daya komputasi. Penelitian ini melibatkan konfigurasi tiga server virtual, di mana masing-masing server secara virtual dilengkapi dengan spesifikasi yang seragam: RAM sebesar 1 GB yang cukup untuk menjalankan aplikasi dasar dan beberapa tugas pengelolaan data, prosesor CPU dengan inti tunggal yang mampu menangani instruksi pemrosesan secara efisien, dan penyimpanan SSD berkapasitas 20 GB yang menawarkan kecepatan akses data yang lebih cepat dibandingkan dengan penyimpanan HDD tradisional.

Perangkat utama yang digunakan dalam penelitian ini adalah laptop dengan spesifikasi AMD Ryzen 5 2500U, prosesor yang dikenal dengan kinerja tinggi dan efisiensi energi. Prosesor ini dilengkapi dengan Radeon Vega Mobile Gfx, unit grafis yang mendukung rendering visual yang lebih baik dan performa komputasi yang optimal. Selain itu, laptop ini didukung oleh memori RAM sebesar 8 GB, yang memberikan kapasitas yang lebih dari cukup untuk mengelola beberapa VM secara simultan tanpa mengalami penurunan kinerja yang signifikan.

Penggunaan VM dalam penelitian ini memungkinkan simulasi lingkungan server yang beragam dan kompleks dalam satu perangkat fisik, memudahkan peneliti untuk menguji berbagai skenario penggunaan dan konfigurasi server tanpa perlu investasi *hardware* tambahan [16]. Hal ini tidak hanya mengurangi biaya operasional tetapi juga meningkatkan efisiensi dalam pengujian dan pengembangan solusi teknologi. Keseluruhan konfigurasi ini bertujuan untuk mengevaluasi performa dan skalabilitas sistem dalam lingkungan yang terkontrol, sehingga memberikan *insight* berharga bagi pengembangan teknologi server dan manajemen data dalam skala yang lebih luas.

III. Hasil dan Pembahasan

Load Balancing merupakan sebuah teknik esensial dalam jaringan komputer yang bertujuan untuk mendistribusikan beban kerja yang masuk ke berbagai server atau kluster server yang berfungsi sebagai satu entitas terpadu. Teknik ini vital untuk memastikan keandalan dan ketersediaan sumber daya dalam sebuah jaringan dengan membagi beban secara merata, sehingga menghindari *overloading* pada server tertentu [17].

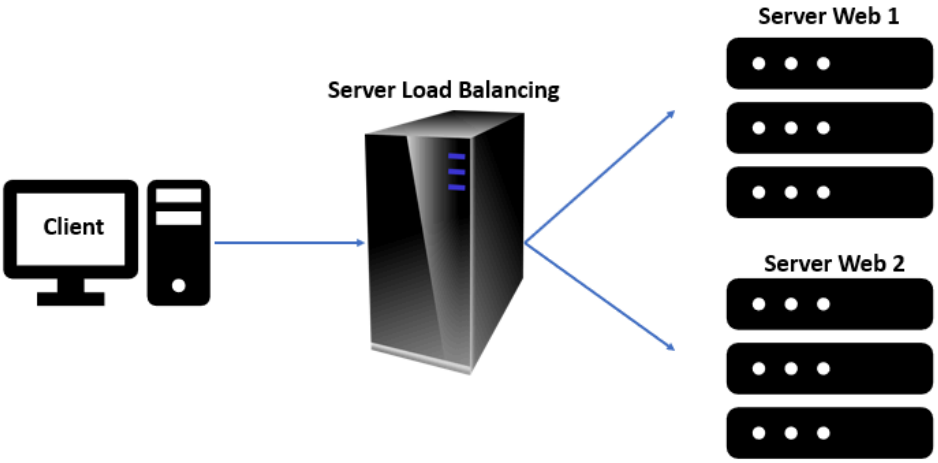
Setelah melakukan serangkaian pengujian komprehensif dan analisis mendalam terhadap sistem server yang telah dikonfigurasi, bagian ini akan mendetailkan hasil yang diperoleh dan membahasnya secara menyeluruh. Hasil pengujian ini disusun untuk mengevaluasi kinerja dari dua algoritma *Load Balancing*, yaitu SHS (*Source Hash Scheduling*) dan URI (*Uniform Resource Identifier*), dengan fokus utama pada parameter *throughput* yang dihasilkan oleh masing-masing algoritma. Parameter ini dipilih karena secara langsung mencerminkan efisiensi algoritma dalam mengelola dan memproses permintaan yang masuk ke server [18].

Analisis kinerja ini bertujuan untuk mengidentifikasi sejauh mana efektivitas dari setiap algoritma dalam meningkatkan performa server web. Ini meliputi penilaian terhadap kecepatan respons server terhadap permintaan pengguna, kemampuan algoritma dalam memaksimalkan penggunaan sumber daya, serta stabilitas dan kehandalan mereka dalam kondisi trafik yang berat. Selain itu, analisis ini juga akan memberikan wawasan yang lebih jelas mengenai kelebihan dan kekurangan dari setiap pendekatan yang diterapkan. Kelebihan

mungkin mencakup kemampuan untuk menangani spike trafik secara efektif, sementara kekurangan bisa berupa kompleksitas konfigurasi atau kebutuhan sumber daya yang lebih tinggi.

Keseluruhan diskusi ini akan didukung oleh data yang dikumpulkan selama fase pengujian, termasuk grafik *throughput*, *log server*, dan *feedback* dari *monitoring tools*. Diskusi ini tidak hanya penting untuk pengembangan lebih lanjut dari sistem yang ada tetapi juga berguna bagi pengambilan keputusan strategis dalam implementasi *Load Balancing* di lingkungan jaringan yang lain. Dengan demikian, hasil dari pengujian ini akan memberikan panduan berharga bagi teknisi dan arsitek jaringan dalam merancang dan mengoptimalkan infrastruktur TI yang efisien dan efektif.

Langkah pertama dalam melakukan pengujian kinerja Load Balancing dengan menggunakan SHS dan URI yangit dengan merancang arsitektur server terlebih dahulu. Arsitektur server inilah nantinya yang digunakan untuk menguji kedua algoritma tersebut. Arsitektur komputer yang dirancang divisualisasikan pada Gambar 4.



Gambar 4. Arsitektur server dengan *Load Balancing* [19]

Gambar 4 memberikan representasi visual dari arsitektur server yang mengintegrasikan teknologi *Load Balancing*. Dalam diagram yang ditampilkan, proses dimulai ketika klien mengirimkan permintaan ke sistem. Permintaan ini pertama kali diterima oleh server *Load Balancing* yang berperan penting dalam arsitektur ini. Server *Load Balancing* memiliki tugas krusial untuk mendistribusikan permintaan yang masuk ke dua server web yang berbeda, yakni Server Web 1 dan Server Web 2. Tujuan dari pendekatan ini adalah untuk memastikan bahwa beban kerja yang diterima oleh setiap server didistribusikan secara merata dan efisien, sehingga meningkatkan kinerja sistem secara keseluruhan. Untuk memverifikasi efektivitas dan efisiensi dari sistem *Load Balancing* ini, dilakukan pengujian berbagai parameter, yang bertujuan untuk menilai kemampuan server dalam menangani berbagai jenis permintaan pengguna selama periode waktu tertentu. Dengan demikian, arsitektur ini dirancang tidak hanya untuk meningkatkan kecepatan respons terhadap permintaan pengguna, tetapi juga untuk meningkatkan stabilitas dan skalabilitas sistem dalam menghadapi beban kerja yang berfluktuasi.

Proses selanjutnya, yaitu menyiapkan perangkat yang digunakan serta konfigurasi Alamat IP untuk setiap perangkat yang digunakan. Identifikasi dan sinkronisasi komunikasi antar perangkat dalam suatu jaringan dimungkinkan melalui penerapan alamat IP yang distingtif untuk setiap perangkat. Perangkat yang digunakan serta Alamat IP untuk mengkonfigurasi perangkat tersebut disusun pada Tabel 1.

Tabel 1. Perangkat dan Alamat IP

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

Tabel 1 menyajikan daftar perangkat yang digunakan dalam sistem jaringan berserta alamat IP masing-masing, yang merupakan komponen kunci dalam pengelolaan dan konfigurasi jaringan. Tabel ini mencakup beberapa elemen penting dalam infrastruktur jaringan, antara lain *load balancer*, Server Web 1, Server Web 2, dan pengguna. Penggunaan alamat IP yang unik untuk setiap perangkat memfasilitasi identifikasi dan koordinasi komunikasi antar perangkat di dalam jaringan. Dalam daftar tersebut, *load balancer* diberikan alamat IP 192.168.109.184, yang bertugas mendistribusikan permintaan ke server-server web untuk mengoptimalkan penggunaan sumber daya dan memastikan distribusi beban yang merata. Server Web 1 tercatat dengan alamat IP 192.168.109.179, sementara Server Web 2 menggunakan alamat IP 192.168.109.178, keduanya melayani permintaan dari pengguna dan memproses data sesuai dengan distribusi yang dilakukan oleh *load balancer*. Pengguna dalam sistem ini terhubung melalui alamat IP 192.168.109.182, yang memungkinkan mereka untuk mengakses layanan yang disediakan oleh server-server tersebut. Informasi ini sangat penting tidak hanya untuk memastikan fungsi komunikasi yang efektif antar perangkat, tetapi juga untuk troubleshooting dan pengelolaan jaringan yang efisien, memungkinkan administrator jaringan untuk mengidentifikasi dan mengatasi potensi masalah dalam alokasi sumber daya atau konflik alamat IP.

Selanjutnya, dilakukan pengujian terhadap algoritma *Load Balancing* menggunakan SHS dan URI. Pengujian ini diimplementasikan pada lima tingkat koneksi berbeda, dengan rentang mulai dari 1000/100 hingga 5000/500, di mana angka tersebut merepresentasikan jumlah permintaan dan koneksi yang dilakukan per detik. Dalam proses pengujian ini, masing-masing tingkat koneksi diuji sebanyak lima kali untuk mendapatkan data yang representatif dan akurat mengenai kinerja kedua algoritma tersebut. Tabel 2 menyajikan hasil uji coba performa dari dua algoritma *Load Balancing*, yaitu SHS dan URI, diukur dalam satuan *kilobyte* per detik (KB/s).

Tabel 2. Hasil pengujian *throughput* pada algoritma SHS dan URI

No	Tingkat Koneksi	SHS (KB/s)					URI (KB/s)				
		Pengujian ke-					Pengujian ke-				
		Satu	Dua	Tiga	Empat	Lima	Satu	Dua	Tiga	Empat	Lima
1	1000/100	51.2	51.2	51.2	51.2	51.2	51.2	51.2	51.2	51.2	51.2
2	2000/200	102.3	102.3	102.3	102.3	102.4	102.4	102.3	102.4	102.3	102.3
3	3000/300	153.5	153.5	153.5	153.5	153.5	153.5	153.5	153.5	153.5	153.1
4	4000/400	204.6	204.6	204.6	204.6	204.7	204.4	204.6	204.6	204.6	204.6
5	5000/500	255.7	255.8	255.8	255.7	255.7	255.8	255.7	255.7	255.7	255.6

Pada Tabel 2, terlihat bahwa pada tingkat koneksi terendah 1000/100, kecepatan transfer data untuk kedua algoritma tercatat stabil pada 51,2 KB/s, menunjukkan kemampuan algoritma dalam menangani permintaan pada skala kecil dengan efisiensi yang tinggi. Seiring dengan peningkatan jumlah koneksi dari 2000/200 hingga mencapai puncaknya pada 5000/500—kecepatan transfer data kedua algoritma tersebut juga menunjukkan peningkatan yang signifikan. Pada koneksi tertinggi, kecepatan transfer mencapai 255.7 KB/s, menandakan bahwa baik SHS maupun URI mampu skala up dengan baik dan mempertahankan efisiensi mereka meskipun beban jaringan meningkat.

Hasil ini menunjukkan bahwa kedua algoritma *Load Balancing* menggunakan SHS dan URI tidak hanya mampu menangani jumlah permintaan dan koneksi yang lebih tinggi, tetapi juga mempertahankan konsistensi performa mereka. Kedua algoritma tersebut efektif dalam mendistribusikan beban permintaan secara merata, yang krusial dalam meminimalisir waktu tunggu dan memaksimalkan *throughput* jaringan. Analisis ini penting dalam membantu pengambilan keputusan teknis untuk pemilihan algoritma yang paling sesuai dalam konfigurasi jaringan yang dinamis dan tuntutan performa yang tinggi.

Berikutnya, berdasarkan pengujian *throughput* yang dilakukan disusun hasil rata-ratanya berdasarkan tingkat koneksi yang diuji mencakup rentang dari 1000/100 hingga 5000/500. Hasil rata-rata untuk uji *throughput* unruk kedau algoritma yang diterapkan tersaji pada Tabel 3.

DOI: http://dx.doi.org/10.22441/fifo.2024.v16i2.008

Tabel 3. Hasil Rata-rata Pengujian *Throughput* (Kbs)

No	Tingkat Koneksi	Hasil Rata-rata Pengujian *Throughput* (KB/s)	
		SHS	URI
1	1000/100	51.20	51.20
2	2000/200	102.32	102.34
3	3000/300	153.50	153.42
4	4000/400	204.62	204.56
5	5000/500	255.74	255.70

Tabel 3 menampilkan hasil pengujian *throughput* rata-rata, diukur dalam *kilobyte* per detik (KB/s), yang diperoleh dari dua algoritma *Load Balancing* yang berbeda, yaitu SHS dan URI, pada lima tingkat koneksi yang berbeda. Tingkat koneksi yang diuji mencakup rentang dari 1000/100 hingga 5000/500, dimana setiap tingkat menggambarkan kombinasi jumlah permintaan dan koneksi yang dilakukan setiap detiknya.

Pada tingkat koneksi awal yaitu 1000/100, kedua algoritma, SHS dan URI, menghasilkan *throughput* yang identik, yakni 51.20 KB/s. Hal ini menunjukkan kemampuan dasar kedua algoritma dalam menangani permintaan pada skala rendah dengan efisiensi yang setara. Seiring dengan meningkatnya tingkat koneksi menjadi 2000/200, SHS mencatat *throughput* rata-rata sebesar 102.32 KB/s, sementara URI menunjukkan hasil yang sangat serupa, hanya sedikit lebih tinggi, yaitu 102.34 KB/s. Perbedaan yang minimal ini mengindikasikan bahwa kedua algoritma memiliki kemampuan pengolahan yang seragam pada tingkat beban ini.

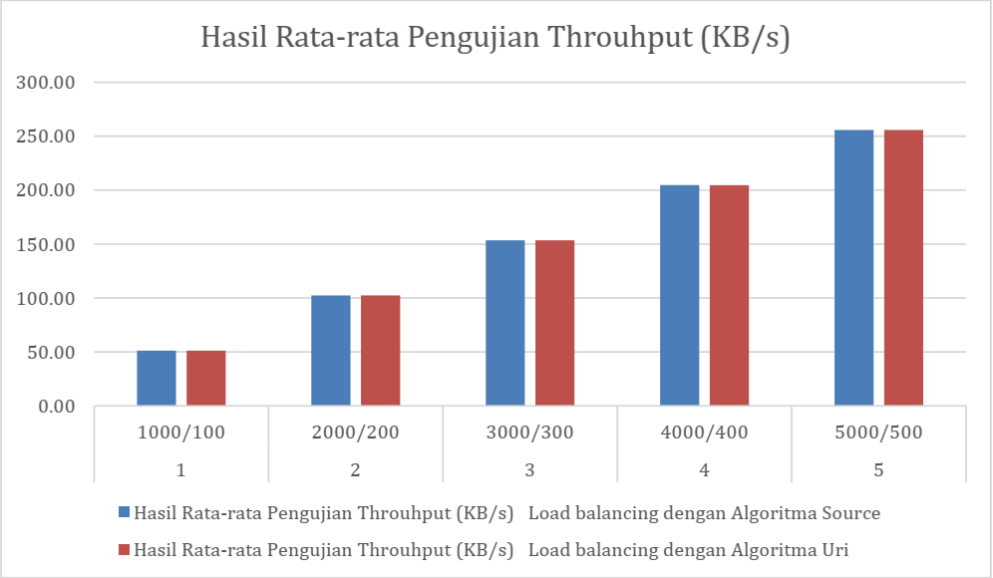
Ketika tingkat koneksi ditingkatkan lagi menjadi 3000/300, SHS menunjukkan *throughput* rata-rata sebesar 153.50 KB/s, sedangkan URI sedikit lebih rendah, dengan 153.42 KB/s. Meskipun ada perbedaan kecil, kedua algoritma tersebut masih menunjukkan performa yang konsisten. Pada tingkat koneksi 4000/400, SHS mencapai *throughput* rata-rata sebesar 204.62 KB/s, sementara URI menghasilkan *throughput* sedikit lebih rendah, yaitu 204.56 KB/s.

Terakhir, pada tingkat koneksi tertinggi, yaitu 5000/500, kedua algoritma menunjukkan kinerja yang hampir identik, dengan SHS mencapai *throughput* 255.74 KB/s dan URI 255.70 KB/s. Hasil ini menegaskan bahwa kedua algoritma tersebut sangat efektif dan serupa dalam pengelolaan dan distribusi beban, bahkan di tingkat koneksi yang paling tinggi.

Secara keseluruhan, tabel ini mengindikasikan bahwa kedua algoritma *Load Balancing*, SHS dan URI, memberikan hasil *throughput* yang sangat mirip di berbagai tingkat koneksi, dengan perbedaan yang sangat kecil dalam hasil rata-rata pengujian. Keserupaan ini menunjukkan bahwa kedua algoritma tersebut telah dioptimalkan untuk performa yang konsisten di berbagai skenario beban jaringan.

Untuk memudahkan dalam menganalisis perbandingan hasil uji algoritma SHS dan URI terhada *throughput* maka dibuat dalam bentuk grafik batang. Grafik yang disajikan menggambarkan rata-rata *throughput* dalam *kilobyte* per detik (KB/s) untuk dua algoritma Load Balancing, SHS dan URI, pada lima tingkat koneksi yang berbeda. Sumbu horizontal pada grafik ini menunjukkan tingkat koneksi yang diuji, mulai dari 1000/100 hingga 5000/500, sedangkan sumbu vertikal menggambarkan *throughput* yang diperoleh dalam KB/s. Dua warna berbeda digunakan dalam grafik untuk membedakan hasil antara kedua algoritma: biru untuk SHS dan merah untuk URI. Perbandingan rata-rata *throughput* untuk metode SHS dan URI dapat dilihat pada Gambar 5.

DOI: http://dx.doi.org/10.22441/fifo.2024.v16i2.008



Gambar 5. Grafik dari *Throughput* (ms)

Pada Gambar 5 menunjukkan bahwa pada tingkat koneksi terendah (1000/100), kedua algoritma menghasilkan *throughput* yang sama, yaitu sekitar 51.2 KB/s. Namun, ketika tingkat koneksi meningkat, grafik menunjukkan bahwa kedua algoritma tetap memperlihatkan hasil *throughput* yang sangat serupa, dengan perbedaan yang minimal pada tingkat koneksi tertentu. Misalnya, pada tingkat koneksi 2000/200, URI terlihat sedikit lebih unggul dengan margin yang sangat tipis, namun pada tingkat 3000/300, SHS memimpin dengan selisih yang hampir tidak terlihat.

Lebih lanjut, pada tingkat koneksi yang lebih tinggi, seperti 4000/400 dan 5000/500, grafik menunjukkan bahwa hasil *throughput* dari kedua algoritma semakin mendekati satu sama lain, dengan perbedaan yang sangat kecil, menandakan performa yang sangat serupa. Hal ini menunjukkan bahwa kedua algoritma tersebut telah dioptimalkan untuk menghadapi peningkatan beban dengan efisiensi yang hampir setara, mempertahankan konsistensi dan reliabilitas dalam menangani permintaan yang bertambah.

Dengan demikian, grafik ini tidak hanya mengilustrasikan konsistensi performa kedua algoritma dalam mempertahankan *throughput* yang stabil di seluruh tingkat koneksi yang diuji, tetapi juga memberikan wawasan penting tentang kemampuan adaptasi dan skala kinerja kedua sistem *Load Balancing* dalam situasi jaringan yang bervariasi. Ini adalah informasi penting bagi para-administrator jaringan dan pengembang sistem dalam mengevaluasi dan memilih teknologi *Load Balancing* yang paling cocok untuk infrastruktur mereka.

Analisis kinerja *throughput* dari kedua algoritma *Load Balancing*, SHS dan URI, mengungkapkan bahwa mereka memiliki performa yang sangat kompetitif dan konsisten di berbagai tingkat koneksi yang diuji. Pada tingkat koneksi yang lebih rendah, seperti pada 1000/100, kedua algoritma tersebut menghasilkan nilai *throughput* yang sama, yakni 51.2 KB/s. Temuan ini mengindikasikan bahwa pada skala kecil, kedua algoritma tersebut mampu mengelola beban dengan efisiensi yang tinggi tanpa adanya perbedaan yang signifikan dalam performa.

Namun, seiring dengan peningkatan tingkat koneksi, dari 2000/200 hingga 5000/500, mulai terlihat perbedaan kecil dalam performa antara kedua algoritma tersebut. Secara umum, URI menunjukkan keunggulan dalam *throughput* pada beberapa tingkat koneksi tertentu, seperti pada 2000/200, sementara SHS memiliki performa yang lebih baik pada tingkat koneksi 3000/300. Pada tingkat koneksi yang lebih tinggi, seperti 4000/400 dan 5000/500, perbedaan *throughput* antara SHS dan URI menjadi sangat minim, hampir tidak terlihat, yang menandakan bahwa kedua algoritma tersebut mampu menjaga kinerja yang stabil dan efektif meskipun beban koneksi meningkat.

Dari analisis yang telah dilakukan, baik SHS maupun URI menunjukkan kemampuan yang solid dalam mengelola *throughput* di berbagai tingkat koneksi. Performa yang konsisten dari kedua algoritma ini membuktikan bahwa mereka dapat diandalkan dalam menghadapi beban jaringan yang beragam. Meskipun kedua algoritma tersebut menunjukkan kinerja yang serupa, URI cenderung memberikan hasil yang sedikit lebih

tinggi dan stabil pada beberapa kondisi, yang mungkin menjadi pertimbangan penting dalam memilih algoritma *Load Balancing*. Ini berarti URI dapat lebih efektif dalam mengelompokkan dan mendistribusikan permintaan yang memiliki karakteristik tertentu, seperti akses terhadap halaman atau layanan tertentu, yang sering kali memiliki pola akses yang berulang. Dengan melakukan distribusi berdasarkan URI, beban kerja dapat disebar lebih merata, mengurangi kemungkinan penumpukan beban pada server tertentu. Pemilihan kedua algoritma akan tergantung pada kebutuhan spesifik dari infrastruktur jaringan yang digunakan serta kriteria keandalan dan efisiensi yang diutamakan oleh administrator jaringan.

IV. Kesimpulan

Dari analisis kinerja *throughput*, dapat disimpulkan bahwa baik SHS maupun URI menunjukkan performa yang sangat baik dan konsisten pada berbagai tingkat koneksi. Pada koneksi rendah, seperti 1000/100, kedua algoritma mencatat *throughput* identik sebesar 51.20 KB/s. Namun, pada tingkat koneksi yang lebih tinggi, seperti 2000/200 hingga 5000/500, URI sedikit lebih unggul dengan *throughput* hingga 255.70 KB/s, dibandingkan dengan 255.74 KB/s pada SHS. Perbedaan performa antara kedua algoritma sangat kecil, tetapi URI cenderung lebih stabil dalam menjaga *throughput*. Pemilihan algoritma dapat didasarkan pada kebutuhan spesifik terkait stabilitas atau performa pada tingkat koneksi tertentu. Kedua algoritma ini menawarkan solusi yang andal untuk meningkatkan kinerja dan kepuasan pengguna server web. Untuk penelitian selanjutnya, ada beberapa aspek yang bisa diteliti lebih lanjut untuk mendapatkan pemahaman yang lebih dalam tentang efektivitas kedua algoritma dalam berbagai skenario operasional. Pertama, penelitian dapat dilakukan untuk menguji kedua algoritma di bawah kondisi jaringan yang lebih bervariasi, termasuk kondisi jaringan yang mengalami gangguan atau fluktuasi yang tinggi. Kedua, penelitian yang menggabungkan metode pengukuran lain seperti latensi dan efisiensi penggunaan sumber daya juga akan berguna untuk mendapatkan gambaran yang lebih komprehensif tentang performa keseluruhan algoritma. Akhirnya, mengimplementasikan algoritma-algoritma ini dalam lingkungan jaringan nyata dan mengukur dampaknya pada pengalaman pengguna akhir dapat memberikan wawasan yang lebih praktis dan relevan bagi pengembang dan pengelola jaringan.

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] S. D. Riskiono and D. Pasha, "Analisis Metode Load Balancing Dalam Meningkatkan Kinerja Website E-Learning," J. Teknoinfo, vol. 14, no. 1, p. 22, 2020, doi: 10.33365/jti.v14i1.466.
[17] 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.
[18] 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.
[19] S. D. Riskiono and D. Darwis, "Peran Load Balancing Dalam Meningkatkan Kinerja Web Server Di Lingkungan Cloud," Krea-TIF, vol. 8, no. 2, p. 1, 2020, doi: 10.32832/kreatif.v8i2.3503.