Jurnal Infomedia : Teknik Informatika, Multimedia, dan Jaringan
Volume 9, Nomor 2, Desember 2024.
ISSN 2527-9858 (online), 2549-1180 (print)
Artikel Penelitian

Implementasi Sistem *Load Balancing* untuk Optimasi Kinerja pada *Web Server Nginx* Menggunakan *Algoritma Ip Hash*

Muhammad Ilham[1], Atthariq[2*], Mursyidah[3]

[1] Program Studi Teknologi Rekayasa Komputer Jaringan, Jurusan Teknologi Informasi dan Komputer, Politeknik Negeri Lhokseumawe, Jln. B.Aceh Medan Km.280, Buketrata, 24301, Indonesia, email: ilham.lsm2018@gmail.com
[2,] Program Studi Teknologi Rekayasa Komputer Jaringan, Jurusan Teknologi Informasi dan Komputer, Politeknik Negeri Lhokseumawe, Jln. B.Aceh Medan Km.280, Buketrata, 24301, Indonesia, email: atthariq.huzaifah@pnl.ac.id
[3,] Program Studi Teknologi Rekayasa Komputer Jaringan, Jurusan Teknologi Informasi dan Komputer, Politeknik Negeri Lhokseumawe, Jln. B.Aceh Medan Km.280, Buketrata, 24301, Indonesia, email: mursyisah@pnl.ac.id
*Corresponding Author: atthariq.huzaifah@pnl.ac.id

Abstrak
Pada era digital saat ini, penggunaan internet semakin meningkat, mengakibatkan lonjakan jumlah pengguna situs web yang dapat menyebabkan server overload. Overload server berpotensi menyebabkan downtime, sehingga diperlukan solusi yang efektif untuk mendistribusikan beban kerja secara merata. Salah satu solusi yang dapat diterapkan adalah menggunakan sistem Load Balancing berbasis Ip Hash. Teknik ini mendistribusikan permintaan *traffic* ke berbagai server berdasarkan data IP pengguna, sehingga beban kerja dapat tersebar dengan lebih merata dan cepat. Penelitian ini bertujuan untuk mengevaluasi efektivitas Load Balancing Ip Hash dalam mengelola beban kerja server web. Uji kinerja dilakukan menggunakan Apache JMeter dengan variasi jumlah *traffic*, yaitu 100, 500, 1000, 2500, dan 5000 permintaan. Hasil menunjukkan bahwa dengan Load Balancing Ip Hash, throughput berkisar antara 32,6 hingga 84,1 transaksi//detik, sedangkan tanpa Load Balancing Ip Hash, Throughput berada di antara 31,9 hingga 72,1 transaksi//detik. Delay dengan Load Balancing Ip Hash konsisten menurun dari 30 ms hingga 2 ms, sementara tanpa Load Balancing Ip Hash, Delay bervariasi dari 36 ms hingga 85 ms. Packet Loss Rate tetap rendah dengan Load Balancing Ip Hash (0,00% hingga 0,02%), dibandingkan tanpa Load Balancing Ip Hash (0,01% hingga 0,05%).
Kata Kunci: **Load Balancing, Ip Hash, Server Web, Apache JMeter, Wireshark, Throughput.**

Abstract
In the current digital era, the increasing use of the internet has led to a surge in website users, which can result in server overload. Server overload has the potential to cause downtime, making it essential to have an effective solution for evenly distributing the workload. One solution that can be implemented is an IP Hash-based Load Balancing system. This technique distributes traffic requests to various servers based on the users' IP data, allowing the workload to be spread more evenly and efficiently. This research aims to evaluate the effectiveness of IP Hash Load Balancing in managing web server workloads. Performance testing was conducted using Apache JMeter with varying traffic loads of 100, 500, 1000, 2500, and 5000 requests. The results show that with IP Hash Load Balancing, throughput ranged from 32.6 to 84.1 transactions per second, while without IP Hash Load Balancing, throughput was between 31.9 and 72.1 transactions per second. The delay with IP Hash Load Balancing consistently decreased from 30 ms to 2 ms, while without IP Hash Load Balancing, the delay varied from 36 ms to 85 ms. Packet loss rates remained low with IP Hash Load Balancing (0.00% to 0.02%) compared to without IP Hash Load Balancing (0.01% to 0.05%)
Keywords: Load Balancing, IP Hash, Web server, Apache JMeter, Wireshark, Throughput.

PENDAHULUAN

Saat ini, internet semakin penting dan semakin banyak orang yang menggunakannya untuk mencari informasi. Selain itu, semakin banyak *Web*site yang dikunjungi oleh pengguna internet untuk mencari berbagai informasi yang mereka butuhkan. Jumlah pengguna situs *Web* terus meningkat dan akan terus meningkat. Kebanyakan situs *Web* dan aplikasi *Web* berjalan lancar selama hanya tiga orang mengunjungi setiap saat.(Magfa et al., 2022) Namun, jika ribuan orang mengakses situs *Web* atau aplikasi pada saat yang sama, *Web server* akan mengalami *overload* karena tidak dapat menampung lebih banyak request pengguna. (Syaqia & Asmunin, 2017) *Overload* akan menyebabkan *server down* karena tidak dapat menjalankan permintaan, sehingga diperlukan sistem Load Balancing Ip Hash untuk mengatasi *server* down(Denny et al., 2021). Sistem Load Balancing Ip Hash dapat bertugas untuk mendistribusikan beban kerja ke banyak *server*, dengan mempertimbangkan kapasitas masing-masing *server*. (Dani & Suryawa, 2017). Ketika permintaan klien telah didistribusikan *sec*ara merata ke setiap node cluster, sistem Load Balancing Ip Hash dapat berfungsi dengan baik. Ini menghindari *overload server* dan memungkinkan *Web server* untuk menerima 10.000 permintaan dengan tidak mengalami

error permintaan. (Rahmatulloh & Msn, 2017). Teknik Load Balancing Ip Hash adalah solusi teknologi yang sangat baik untuk memaksimalkan bandwith internet tanpa ketimpangan. (Warman & Andrian, 2017)

Beberapa penelitian yang telah dilakukan untuk penerapan Teknik Load Balancing Ip Hash menggunakan beberapa metode memberikan hasil keseimbangan *Traffic* yang berbeda beda. Implementasi sistem Load Balancing Ip Hash *Web server* pada jaringan *public cloud computing* menggunakan *least connection*, Ditunjukkan bahwa penerapan sistem pengimbangan beban sudah sesuai dengan metode *least connection*, yang membagi beban berdasarkan banyaknya koneksi yang dilayani oleh *server*. (Fadila et al., 2023). Pada penelitian yang berjudul Load Balancing Ip Hash dengan metode *round robin* untuk pembagian beban kerja *Web server*, Hasil yang diperoleh menunjukkan bahwa metode Load Balancing Ip Hash *round robin* pada *Nginx* yang terkoneksi dengan dua mesin Apache *Web server* dapat membantu mengatur pembagian beban kerja *Web server*. (Komaruddin et al., 2019)

Namun, apakah kedua teknik tersebut efektif untuk Load Balancing *Ip Hash*. Peneliti ingin menerapkan metode Load Balancing yang berbeda seperti *Ip Hash*. Metode ini menggunakan data paket masuk IP pengguna untuk mendistribusikan permintaan *traffic* masuk.

METODE PENELITIAN

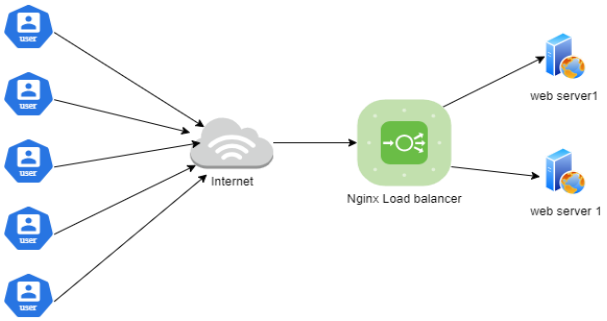
Metode yang digunakan sebagai pengembangan sistem dalam penelitian ini menggunakan metode *Ip Hash*, Algoritma *Ip Hash* Load Balancing adalah sebuah metode untuk mendistribusikan permintaan (request) dari client *sec*ara merata ke beberapa node atau *server* backend. Algoritma ini memainkan peran penting dalam mengelola *traffic* jaringan dan memastikan pemerataan beban jaringan di beberapa *server*.

A. Data dan Pengumpulan Data

Dalam Pengimplementasian Sistem Load Balancing untuk Optimasi Kinerja pada *Web server Nginx* Menggunakan *Algoritma Ip Hash.* Metode Pengumpulan data yang digunakan dalam penelitian ini berupa data primer, yaitu data yang diperoleh dari hasil pengukuran langsung dari peneliti. Data primer yang akan dikumpulkan dalam penelitian ini nantinya adalah data yang telah diuji peneliti mengenai nilai *Throughput, Delay, Packet Loss Rate* saat beroperasi (Hasbi & Saputra, 2021).

B. Rancangan Sistem

Dalam penelitian ini, pengujian terhadap sistem yang nantinya akan diterapkan membutuhkan rancangan agar pengujian dapat berjalan sesuai alur sistem yang jelas, oleh karena itu berikut perancangan sistem yang akan dilakukan di dalam penelitian ini dapat dilihat pada **Gambar 1.**



Gambar 1. Perancangan Sistem Load Balancing Ip Hash

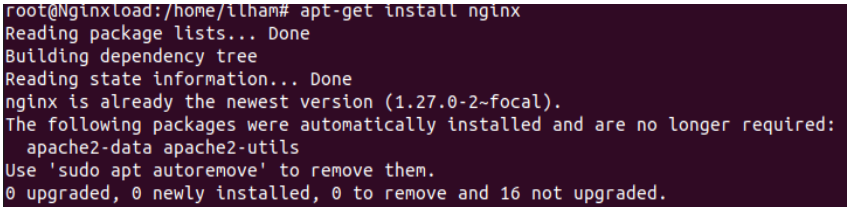
Dalam Ilustrasi, terdapat beberapa user yang akan mengakses aplikasi *web*. Lalu, terdapat 2 buah *web server* (misalnya *Web server* 1, dan *Web server* 2) yang menghosting aplikasi web tersebut dengan konten dan konfigurasi yang identik (Saputra et al., 2023). Kedua web server kemudian dibalik *load balancer*. *Load balancer* bertugas menerima semua permintaan dari klien dan mendistribusikannya ke dua *web server secara* bergantian. Misalnya klien pertama kali mengakses aplikasi web, maka permintaan tersebut kan diarahkan load balancer ke *Web server* 1. Permintaan kedua akan diarahkan ke *Web server* 2. Begitu seterusnya bergantian. Dengan demikian, kedua *web server* tersebut tidak akan *overload* karena permintaan klien didistribusikan *sec*ara merata oleh load balancer. Sisi klien juga akan merasakan performa yang optimal karena permintaan disebar ke beberapa *server* sehingga tidak terjadi hambatan pada satu *server* saja.

C. Teknik Pengujian

Dalam implementasi penelitian ini, peneliti akan menerapkan teknik pengujian yang akan digunakan nantinya berkaitan dengan pengujian jaringan menyelidiki dan mengukur menggunakan *Quality of Service (QOS)* dengan parameter *Throughput, Delay, Packet Loss Rate*.(Saputra et al., 2023) Baik tidaknya layanan jaringan dapat diukur dengan metode kualitas layanan jaringan atau *Quality of Service (QOS)* dan merupakan suatu usaha untuk mendefinisikan karakteristik dan sifat dari suatu layanan jaringan.

D. Menginstal Load Balancing Ip Hash Nginx

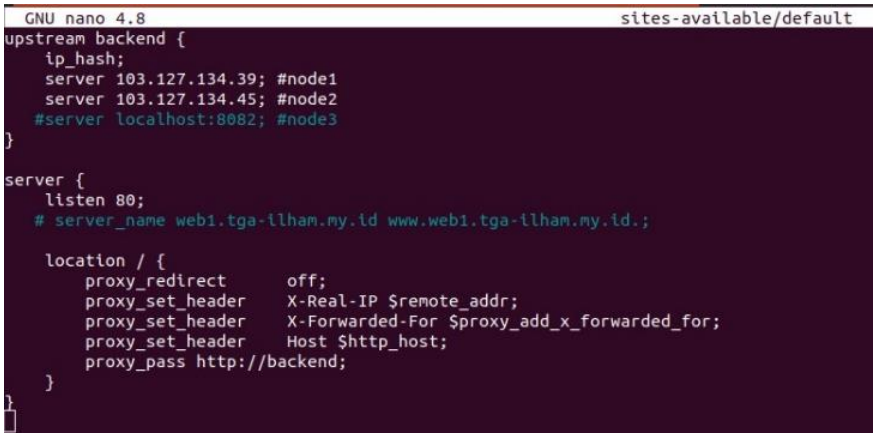
Nginx adalah *server web* yang juga dapat berfungsi sebagai *reverse proxy*, *load balancer*, dan HTTP *cache*. Kemampuan Nginx untuk menangani sejumlah besar koneksi *sec*ara bersamaan membuatnya ideal untuk digunakan sebagai *load balancer* (Dani & Suryawa, 2017). Nginx mendukung beberapa metode Load Balancing dan dapat dikonfigurasi untuk mendistribusikan *traffic sec*ara adil dan cepat ke *server backend*. Instalasi *web server* Nginx seperti pada **Gambar 2.**



Gambar 2. Menginstal *web server* Nginx di Load Balancing

E. Konfigurasi Metode Ip Hash

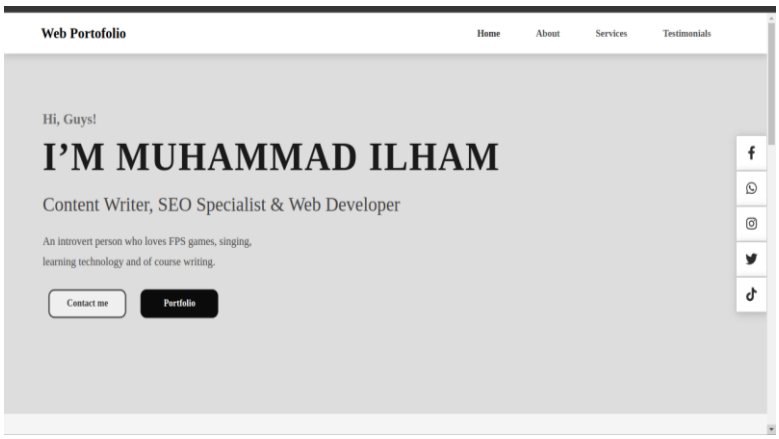
Metode Ip Hash adalah salah satu teknik Load Balancing yang digunakan oleh Nginx untuk mendistribusikan *traffic* ke *server* backend. Dalam metode ini, Nginx menggunakan hash dari alamat IP klien untuk menentukan *server* backend mana yang akan menerima permintaan. Tujuannya adalah untuk memastikan bahwa permintaan dari klien yang sama selalu diarahkan ke *server* yang sama, selama *server* tersebut tersedia. Metode ini sangat berguna untuk menjaga konsistensi sesi pengguna. Untuk konfigurasi Ip Hashnya seperti pada **Gambar 3**.



Gambar 3. Konfigurasi Ip Hash

F. Tampilan Web server

Web server menyediakan layanan untuk menerima permintaan dari *client*. Untuk menerapkan sistem Load Balancing maka dibutuhkan 2 Jalur *web server*. Tampilan *web server* seperti pada **Gambar 4**.



Gambar 4. Tampilan *Web server*

HASIL DAN PEMBAHASAN

A. Pengujian Load Balancing Ip Hash di Jmeter

Pengujian menggunakan Apache JMeter bertujuan untuk mengevaluasi kinerja aplikasi web, server, dan jaringan. Utamanya, JMeter menilai kinerja aplikasi di bawah beban tertentu untuk memastikan waktu respons yang optimal, *Throughput* tinggi, dan keandalan ketika banyak pengguna mengaksesnya. Pengujian beban dan stres menentukan batas kapasitas dan stabilitas aplikasi di bawah kondisi ekstrem. Pengujian skalabilitas dan ketahanan mengukur kinerja saat sumber daya ditambahkan dan di bawah beban jangka panjang. Pengujian volume mengevaluasi kemampuan menangani data besar, sementara pengujian keandalan memastikan aplikasi tetap berfungsi dengan baik. Pengujian fungsional dan regresi memverifikasi semua fungsi aplikasi setelah pembaruan, dan pengujian kapasitas menentukan jumlah maksimum pengguna yang dapat ditangani. JMeter dipilih karena *open source*, *multi-platform*, mendukung berbagai protokol, dan dapat diperluas dengan plugin dan skrip, menjadikannya alat yang ideal untuk memastikan aplikasi menangani trafik dengan cepat dan andal. Hasil pengujian Jmeter menggunakan Load Balancing *Ip Hash*. Pada pengujian di Jmeter, menggunakan tingkat beban 100, 500, 1000, 2500,

dan 5000 pengguna dalam pengujian kinerja bertujuan untuk mengevaluasi aplikasi pada berbagai kondisi penggunaan, dari beban rendah hingga ekstrem. Pengujian ini membantu mengidentifikasi batas kapasitas, mengukur performa, dan mendeteksi hambatan pada setiap tingkat beban. Dengan demikian, bisa memastikan bahwa aplikasi tetap stabil, dapat diskalakan, dan andal bahkan dalam kondisi trafik yang sangat tinggi. Pemilihan angka-angka tersebut memungkinkan pemahaman yang lengkap tentang bagaimana aplikasi berperforma di berbagai skenario, sehingga memberikan kepercayaan diri dalam penggunaan aplikasi secara luas.

1. Jumlah Traffic 100

Melakukan Uji performa dari *web server* Menggunakan Load Balancing Ip Hash dan tanpa Load balancing Ip Hash dengan Traffic 100 permintaan yang akan dilayani oleh *server*. Jika aplikasi Jmeter dijalankan, mendapatkan hasil seperti **Tabel 1.**

Table 1. Hasil Pengujian Dengan Traffic 100

Label	Server	Samples	Average	Error%	*Throughput*
HTTP Request	Load Balancing	100	86	0.00%	84,1/sec
HTTP Request	Tanpa load Balancing	100	71	0,00%	72,1/sec

Pada pengujian dengan 100 sampel, performa server diuji baik dengan maupun tanpa penggunaan load balancing. Hasil menunjukkan bahwa tanpa menggunakan load balancing, server memiliki waktu respon yang lebih cepat, yaitu 71 ms, dibandingkan dengan penggunaan load balancing yang mencatat waktu respons 86 ms. Namun, meskipun waktu respons lebih tinggi pada konfigurasi load balancing, throughput yang dihasilkan lebih besar, yakni 84,1 request per detik, dibandingkan dengan 72,1 request per detik pada konfigurasi tanpa load balancing. Hal ini menunjukkan bahwa, pada beban rendah, penggunaan load balancing masih menghasilkan throughput yang lebih tinggi, meskipun waktu respons sedikit meningkat.

2. Jumlah *Traffic* 500

Melakukan uji performa dari *web server* menggunakan Load Balancing Ip Hash dan tanpa Load Balancing Ip Hash dengan jumlah *Traffic* 500 yang akan dilayani oleh *server* dalam satu waktu. Jika aplikasi Jmeter dijalankan, mendapatkan hasil seperti **Tabel 2** .

Tabel 2. Hasil Pengujian Dengan *Traffic* 500

Label	Server	Samples	Average	Error%	*Throughput*
HTTP Request	Load Balancing	500	106	0.00%	72,2/sec
HTTP Request	Tanpa Load Balancing	500	175	0,00%	64,7/sec

Pada pengujian dengan 500 sampel, performa load balancing mulai menunjukkan hasil yang lebih baik dibandingkan tanpa load balancing. Pada konfigurasi load balancing, rata-rata waktu respons tercatat sebesar 106 ms, sedangkan tanpa load balancing waktu respon meningkat menjadi 175 ms. Dari segi throughput, load balancing mampu menangani 72,2 request per detik, sedangkan tanpa load balancing hanya mampu menangani 64,7 request per detik. Perbedaan ini mengindikasikan bahwa pada jumlah request yang lebih tinggi, load balancing mampu mendistribusikan beban dengan lebih efisien, sehingga tidak hanya menurunkan waktu respon tetapi juga meningkatkan throughput dibandingkan dengan konfigurasi tanpa load balancing.

3. Jumlah *Traffic* 1000

Melakukan uji performa dari *web server* menggunakan Load Balancing Ip Hash dengan jumlah *Traffic* 1000 permintaan yang akan dilayani oleh *server* dalam satu waktu. Jika aplikasi Jmeter dijalankan, mendapatkan hasil seperti **Tabel 3**.

Tabel 3 Hasil Pengujian Dengan *Traffic* 1000

Label	Server	Samples	Average	Error%	*Throughput*
HTTP Request	Load Balancing	1000	141	0.00%	186.0 /sec
HTTP Request	Tanpa load Balancing	1000	234	0.00%	113,3/sec

Pada pengujian dengan 1000 sampel, perbedaan performa antara konfigurasi dengan dan tanpa load balancing semakin terlihat. Pada konfigurasi load balancing, rata-rata waktu respons tercatat sebesar 141 ms, sementara pada konfigurasi tanpa load balancing, waktu respons meningkat secara signifikan menjadi 234 ms. Dari segi throughput, load balancing mampu menangani 186,0 request per detik, jauh lebih tinggi dibandingkan konfigurasi tanpa load balancing yang hanya mencapai 113,3 request per detik. Data ini memperkuat kesimpulan bahwa load balancing semakin efektif dalam mengoptimalkan kinerja server ketika beban permintaan tinggi. Distribusi beban kerja yang lebih merata melalui load balancing memungkinkan server merespons permintaan lebih cepat dan menangani lebih banyak permintaan per detik.

4. Jumlah *Traffic* 2500

Melakukan uji performa dari *web server* menggunakan Load Balancing Ip Hash dengan jumlah *Traffic* 2500 permintaan yang akan dilayani oleh *server* dalam satu waktu. Jika aplikasi Jmeter dijalankan, mendapatkan hasil seperti **Tabel 4.**

Tabel 4 Hasil Pengujian Dengan *Traffic* 2500

Label	Server	Samples	Average	Error%	*Throughput*
HTTP Request	Load Balancing	2500	169	0.00%	162,6/sec
HTTP Request	Tanpa load Balancing	2500	198	0,00%	122,6/sec

Pada pengujian dengan 2500 sampel, hasil menunjukkan bahwa penggunaan load balancing memberikan peningkatan signifikan dalam performa server dibandingkan dengan tanpa load balancing. Konfigurasi load balancing menghasilkan rata-rata waktu respons sebesar 169 ms, lebih cepat daripada konfigurasi tanpa load balancing yang mencatat waktu respons sebesar 198 ms. Selain itu, throughput pada konfigurasi load balancing mencapai 162,6 *request* per detik, lebih tinggi dibandingkan 122,6 *request* per detik pada konfigurasi tanpa load balancing. Data ini mengindikasikan bahwa pada beban yang sangat tinggi, load balancing mampu mendistribusikan beban dengan lebih efisien, sehingga dapat mengurangi waktu respons dan meningkatkan jumlah request yang dapat ditangani per detik. Hal ini semakin menegaskan bahwa load balancing memberikan keuntungan besar dalam hal skalabilitas dan efisiensi kinerja server pada kondisi permintaan yang tinggi.

5. Jumlah *Traffic* 5000

Melakukan uji performa dari *web server* menggunakan Load Balancing Ip Hash dengan jumlah *Traffic* 5000 permintaan yang akan dilayani oleh *server* dalam satu waktu. Jika aplikasi Jmeter dijalankan, mendapatkan hasil seperti **Tabel 5**.

Tabel 5. Hasil Pengujian Load Balancing Dengan *Traffic* 5000

Label	Server	Samples	Average	Error%	*Throughput*
HTTP Request	Load Balancing	5000	210	0,00%	32,6/sec
HTTP Request	Tanpa load Balancing	5000	540	0,00%	31,9/sec

Pada pengujian dengan 5000 sampel, performa server menunjukkan perbedaan yang semakin signifikan antara konfigurasi dengan dan tanpa load balancing. Pada konfigurasi load balancing, rata-rata waktu respons tercatat sebesar 210 ms, yang jauh lebih cepat dibandingkan dengan waktu respons sebesar 540 ms pada konfigurasi tanpa load balancing. Meskipun throughput hanya sedikit lebih tinggi pada konfigurasi load balancing (32,6 request per detik) dibandingkan dengan konfigurasi tanpa load balancing (31,9 request per detik), perbedaan dalam waktu respons yang sangat besar menunjukkan bahwa load balancing mampu menstabilkan waktu respons server saat beban sangat tinggi. Dengan demikian, hasil ini mengonfirmasi bahwa load balancing secara efektif mengurangi waktu respons secara signifikan ketika menangani jumlah permintaan yang besar, meskipun peningkatan throughput relatif kecil pada tingkat beban ini.

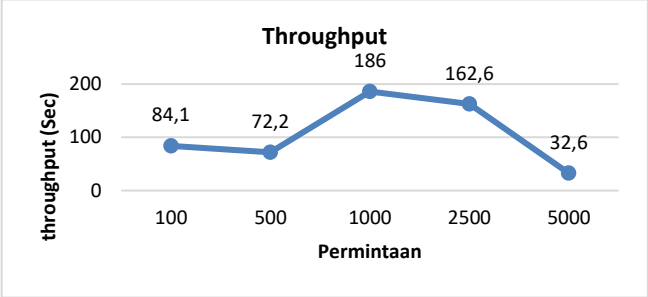
B. Perbandingan Hasil pengujian Jmeter

Hasil Load Balancing Ip Hash dan tanpa Load Balancing Ip Hash dengan menggunakan *Jmeter* dapat dilihat seperti **Tabel 6.**

Tabel 6. Hasil Pengujian Menggunakan Load Balancing Ip Hash

Traffic	*Throughput*	Berhasil
100	84,1/sec	100%
500	72,2/sec	100%
1000	186.0 /sec	100%
2500	162,6/sec	100%
5000	32,6/sec	100%

Jika seluruh nilai dari pengujian dengan *Traffic* yang berbeda dari Tabel di atas dibuat perbandingan maka akan terlihat penurunan atau peningkatan nilai. Untuk melihat perbandingan nilai *Throughput* dari tiap tiap *traffic* dapat dilihat pada **Gambar 5.**



Gambar 5. Diagram Garis Menggunakan Load Balancing Ip Hash

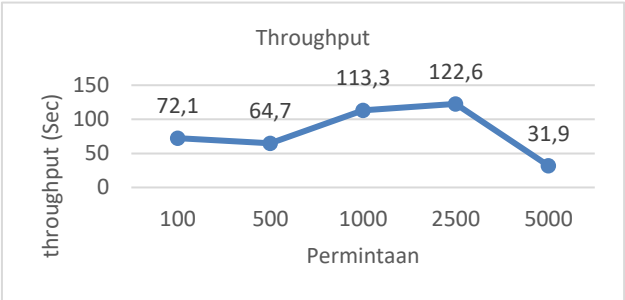
Grafik *Throughput* di atas menggambarkan bagaimana waktu yang dibutuhkan untuk memproses permintaan (*Throughput*) berubah seiring dengan peningkatan jumlah permintaan yang diterima oleh sistem. Pada awalnya, dengan jumlah permintaan sebesar 100, *Throughput* tercatat sekitar 84,1//detik. Ketika jumlah permintaan meningkat menjadi 500, *Throughput* menurun menjadi 72,2//detik, menunjukkan bahwa sistem bekerja lebih cepat dalam menangani permintaan yang lebih besar. Namun, pada titik 1000 permintaan, *Throughput* sedikit meningkat menjadi 84,1//detik, yang mungkin menunjukkan adanya batasan atau faktor tertentu yang memengaruhi kinerja sistem pada tingkat permintaan tersebut. Seiring dengan peningkatan permintaan hingga 2500, *Throughput* kembali menurun menjadi 62,6//detik, dan terus menurun hingga 32,6//detik saat permintaan mencapai 5000. Penurunan ini menunjukkan bahwa sistem semakin cepat dalam memproses permintaan dengan meningkatnya beban, meskipun terdapat sedikit peningkatan waktu proses di tengah-tengah rentang permintaan. Secara keseluruhan, grafik ini memperlihatkan bahwa sistem mampu menangani beban permintaan yang lebih besar dengan waktu pemrosesan yang lebih singkat, meskipun ada perubahan kecil pada titik tertentu.

Hasil tanpa Ip Hash Load Balancing dengan menggunakan jmeter dapat di lihat seperti **Tabel 7**.

Tabel 7. Hasil Tanpa Load Balancing Ip Hash

Traffic	***Throughput***	**Berhasil**
100	72,1/sec	100%
500	64,7/sec	100%
1000	113,3/sec	100%
2500	122,6/sec	100%
5000	31,9/sec	100%

Jika seluruh nilai dari pengujian dengan *Traffic* yang berbeda dari Tabel di atas dibuat perbandingan maka akan terlihat penurunan atau peningkatan nilai. Untuk melihat perbandingan nilai *Throughput* dari tiap tiap *Traffic*, dapat dilihat pada **Gambar 6.**



Gambar 6. Diagram Garis Menggunakan Load Balancing Ip Hash

Grafik Throughput menunjukkan perubahan kinerja sistem dengan dan tanpa Load Balancing Ip Hash seiring peningkatan permintaan. Pada awalnya, Throughput menurun saat permintaan meningkat hingga 500, namun kemudian naik signifikan pada 1000 dan 2500, sebelum menurun drastis pada 5000. Dengan Load Balancing, variasi Throughput lebih besar, sedangkan tanpa Load Balancing, kinerja lebih stabil pada permintaan rendah hingga menengah. Secara keseluruhan, kedua sistem menunjukkan pola peningkatan kinerja pada permintaan besar, meskipun Load Balancing memperlihatkan fluktuasi yang lebih tajam.

C. Pengujian Load Balancing Ip Hash di Wireshark

Pengujian Load Balancing Ip Hash dengan Wireshark bertujuan menganalisis distribusi beban jaringan dan memastikan *traffic* didistribusikan merata ke server berdasarkan Ip Hash. Wireshark membantu menangkap paket, memeriksa apakah header IP dihash dan dialokasikan dengan benar, serta mendeteksi anomali seperti distribusi beban yang tidak merata atau hambatan kinerja. Pengujian ini memverifikasi implementasi dan memungkinkan identifikasi area yang perlu dioptimalkan. Dengan Jmeter, pengujian menggunakan berbagai tingkat beban (100-5000 pengguna) bertujuan

mengevaluasi performa aplikasi di kondisi trafik rendah hingga ekstrem, memastikan aplikasi tetap stabil dan andal.

D. Perbandingan *Quality of Service* di Wireshark

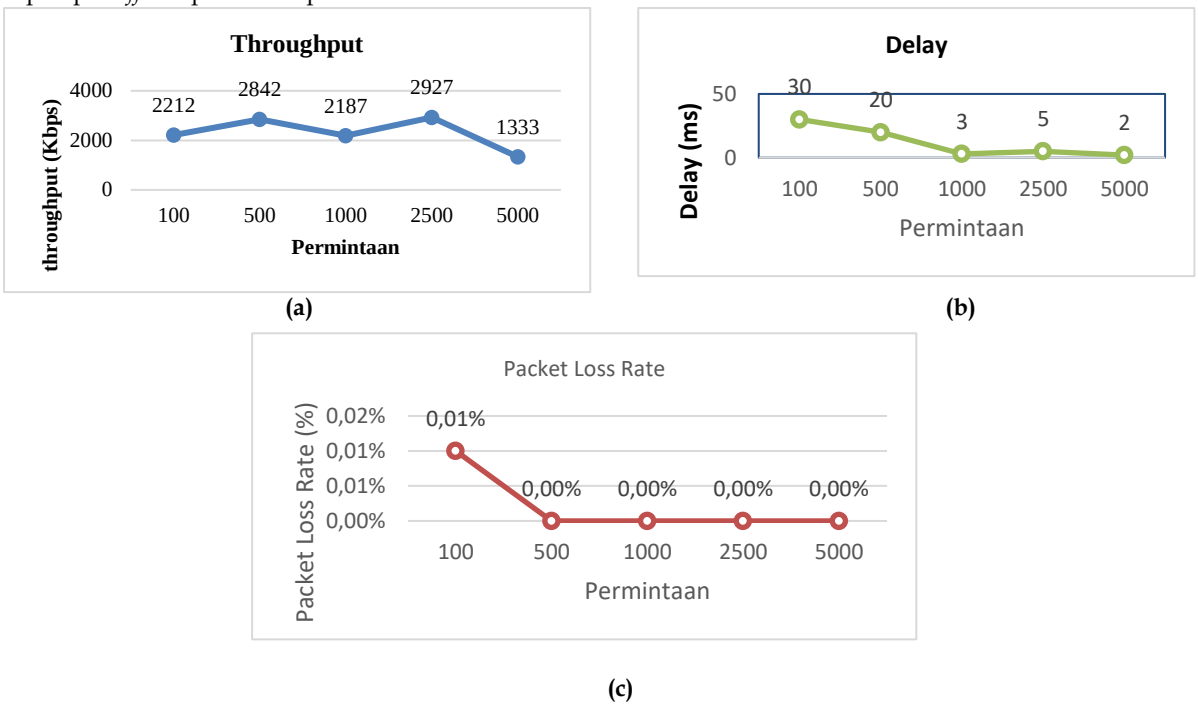
1. Hasil *Qualitiy Of Service* dengan Load Balancing Ip Hash

Hasil *Quality of Service* di Wireshark dengan Load Balancing Ip Hash dapat dilihat seperti **Tabel 8.**

Tabel 8. Hasil Pengujian Load Balancing Ip Hash Di Wireshark

Traffic	*Throughput*	*Delay*	*Packet Loss Rate*
100	2212 Kbps	30 *ms*	0,02%
500	2842 Kbps	20 *ms*	0,00%
1000	2187 Kbps	3 *ms*	0,00%
2500	2927 Kbps	5 *ms*	0,00%
5000	1333 Kbps	2 *ms*	0,00%

Jika seluruh nilai dari pengujian dengan *Traffic* yang berbeda dari Tabel di atas dibuat perbandingan maka akan terlihat penurunan atau peningkatan nilai. Untuk melihat perbandingan nilai *Throughput*, *Delay* dan *Packet Loss Rate* dari tiap tiap *Traffic* dapat dilihat pada **Gambar 7.**



Gambar 7. Diagram garis dengan Load Balancing Ip Hash (a) Throughput (b) Delay (c) Packet Loss rate.

Grafik throughput menunjukkan perubahan laju data yang dihasilkan oleh sistem (dalam satuan Kbps) seiring dengan peningkatan jumlah permintaan. Pada awalnya, *throughput* meningkat dari 2212 Kbps (1 permintaan) hingga mencapai 2927 Kbps pada 100 permintaan, menunjukkan kemampuan sistem dalam menangani permintaan yang lebih besar. Namun, setelah titik ini, *throughput* menurun drastis menjadi 1333 Kbps pada 1000 permintaan, mengindikasikan kesulitan sistem dalam menangani beban kerja yang sangat tinggi. Grafik *delay* juga menunjukkan hubungan antara jumlah permintaan dan latensi dalam skenario load balancing menggunakan algoritma IP Hash. Pada awalnya, *delay* menurun saat permintaan meningkat dari 100 hingga 1000, kemungkinan karena efisiensi distribusi beban yang membaik. Namun, saat permintaan mencapai 2500, *delay* sedikit meningkat akibat beban tambahan pada server, sebelum akhirnya menurun lagi hingga mencapai 5000 permintaan, kemungkinan berkat penyesuaian sistem atau aktivasi server tambahan. Grafik packet loss menunjukkan bahwa seiring dengan peningkatan jumlah permintaan, tingkat kehilangan paket awalnya menurun signifikan dan kemudian stabil pada tingkat yang sangat rendah (0,00%), menunjukkan bahwa sistem load balancing dengan algoritma IP Hash mampu menangani tambahan beban dengan baik. Secara keseluruhan, performa jaringan optimal pada beban sedang, namun menurun pada beban tinggi dengan penurunan throughput dan peningkatan delay meskipun tingkat *packet loss* tetap minimal. Peningkatan lebih lanjut diperlukan untuk menjaga kestabilan performa jaringan pada beban yang sangat tinggi, misalnya melalui optimasi algoritma load balancing atau penambahan kapasitas *server*.

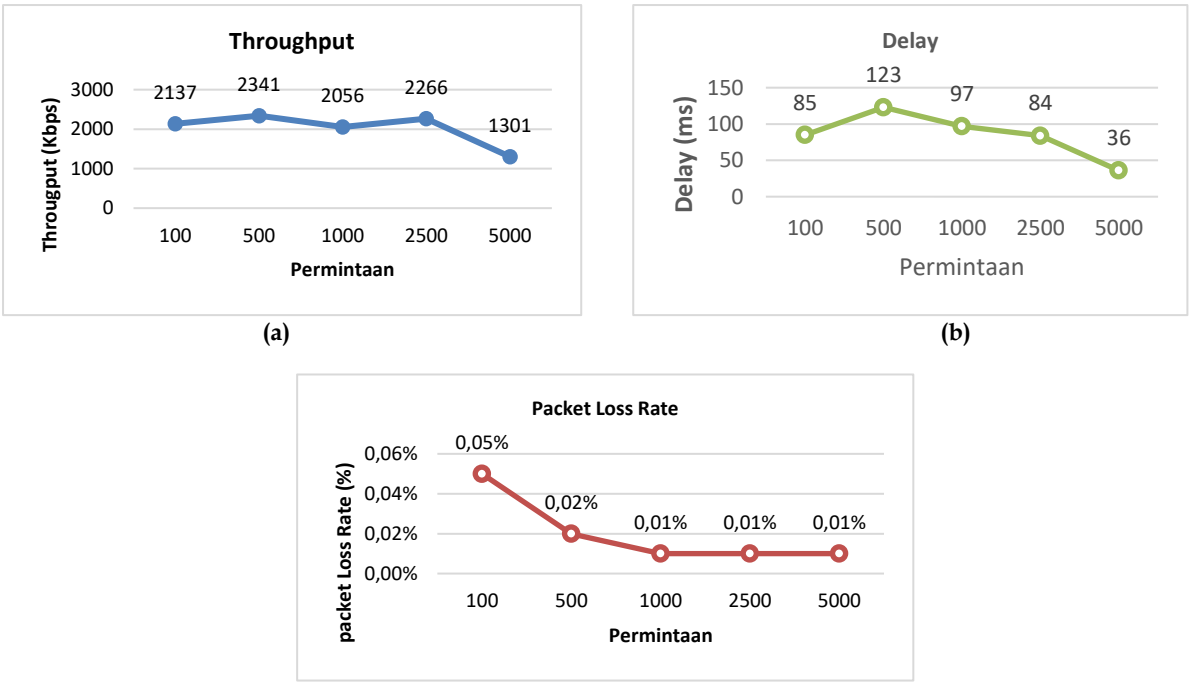
2. Hasil *Quality Of Service* Tanpa Load Balancing ip Hash

Hasil *Quality of Service* di wireshark Tanpa Load Balancing Ip Hash dengan *Traffic* 1 seperti **Tabel 9.**

Tabel 9. Hasil Pengujian Tanpa Load Balancing Di Wireshark

Traffic	*Throughput*	*Delay*	*Packet Loss Rate*
100	2137 Kbps	85 *ms*	0,05%
500	2341 Kbps	123 *ms*	0,02%
1000	2056 kbps	97 *ms*	0,01%
2500	2266 Kbps	84 *ms*	0,01%
5000	1301 Kbps	36 *ms*	0,01%

Jika seluruh nilai dari pengujian dengan *Traffic* yang berbeda dari *Tabel* di atas dibuat perbandingan maka akan terlihat penurunan atau peningkatan nilai. Untuk melihat perbandingan nilai *Throughput*, *Delay*, *Packet Loss Rate* dari tiap tiap *Traffic*, dapat dilihat pada **Gambar 8**.



Gambar 8. Diagram Garis Tanpa Load Balancing Ip Hash (a) Throughput (b) Delay (c) Packet Loss Rate

Grafik throughput menunjukkan bahwa laju data sistem meningkat seiring bertambahnya permintaan, mencapai nilai puncak 2341 Kbps pada 500 permintaan. Namun, setelah titik tersebut, throughput mulai menurun, dengan penurunan tajam pada 5000 permintaan, di mana throughput turun drastis menjadi 1301 Kbps. Hal ini mengindikasikan adanya keterbatasan sistem dalam menangani beban yang sangat tinggi, kemungkinan karena keterbatasan sumber daya atau efisiensi algoritma load balancing yang menurun pada kondisi ini. Grafik delay menunjukkan bahwa waktu tunda meningkat dari 85 ms pada 100 permintaan hingga puncaknya 123 ms pada 500 permintaan, yang mungkin disebabkan oleh peningkatan beban sistem. Setelah 500 permintaan, delay justru menurun meskipun permintaan bertambah, mencapai 36 ms pada 5000 permintaan. Penurunan ini dapat mengindikasikan bahwa sistem menjadi lebih efisien dalam menangani beban yang lebih besar, mungkin karena adanya optimasi atau distribusi beban yang lebih baik sehingga tambahan permintaan tidak memperbesar waktu tunda. Grafik packet loss rate menunjukkan bahwa tingkat kehilangan paket awalnya berada pada 0,05% pada 100 permintaan, kemudian menurun menjadi 0,01% pada 500 dan 1000 permintaan, dan mencapai 0% pada 2500 permintaan. Namun, pada 5000 permintaan, tingkat kehilangan paket kembali naik menjadi 0,05%, yang mengindikasikan adanya kemacetan jaringan pada beban yang sangat tinggi. Secara keseluruhan, hasil ini menunjukkan bahwa sistem dan jaringan memiliki performa optimal pada beban sedang, namun perlu peningkatan lebih lanjut untuk menjaga performa yang konsisten pada beban yang sangat tinggi, seperti optimasi algoritma load balancing atau peningkatan kapasitas sumber daya.

Analisis keseluruhan dari ketiga grafik ini menunjukkan bahwa jaringan berkinerja baik di bawah beban sedang tetapi menghadapi tantangan dalam mempertahankan cepat saat beban meningkat. *Throughput* mencapai puncaknya dan kemudian menurun, menunjukkan titik cepat maksimum diikuti oleh penurunan kinerja karena kemungkinan kemacetan. *Delay* awalnya menurun, tetapi melonjak pada titik beban kritis dan kemudian berkurang lagi, menyoroti area di mana jaringan mengalami kemacetan signifikan. Kehilangan paket minimal dan terkendali dengan baik pada beberapa titik, tetapi peningkatan kehilangan paket pada beban sangat tinggi menunjukkan perlunya peningkatan dalam menangani beban yang lebih tinggi untuk memastikan kinerja jaringan yang konsisten.

Dari pengujian yang dilakukan menggunakan Wireshark pada sistem dengan dan tanpa Load Balancing

menggunakan metode Ip Hash, dapat disimpulkan beberapa hal. Tanpa Load Balancing, *Throughput* meningkat dengan bertambahnya jumlah paket hingga titik tertentu (50 paket) namun kemudian menurun. Sementara itu, dengan Load Balancing, *Throughput* relatif lebih tinggi pada jumlah paket yang lebih besar, menunjukkan cepat yang lebih baik dalam mengelola beban *traffic* jaringan.

Dalam hal *Delay*, tanpa Load Balancing, *Delay* cukup berperubahan namun relatif lebih rendah pada jumlah paket yang kecil (50 dan 1000 paket). Sebaliknya, dengan Load Balancing, *Delay* meningkat signifikan pada jumlah paket yang besar (2500 paket), namun lebih terkontrol pada jumlah paket yang lebih kecil.

Untuk packet loss rate, tanpa Load Balancing, *Packet Loss Rate* hanya signifikan pada 100 paket dan nol untuk jumlah paket yang lebih tinggi. Dengan Load Balancing, *Packet Loss Rate* tetap rendah (0.01%) pada jumlah paket yang lebih kecil (500 dan 1000), namun ada peningkatan pada jumlah paket yang besar (5000 paket).

KESIMPULAN

Berdasarkan hasil dan pembahasan pada penelitian yang telah di lakukan, maka dapat disimpulkan bahwa Pengujian performa jaringan web server setelah diterapkannya load balancing IP hash menunjukkan hasil throughput antara 1333 Kbps hingga 2212 Kbps. Delay dengan sistem load balancing berada pada rentang 2 ms hingga 30 ms, sementara Packet Loss Rate berada di antara 0,00% hingga 0,02%. Perbandingan performa dengan dan tanpa load balancing IP hash juga memperlihatkan variasi throughput, di mana dengan load balancing throughput berkisar antara 84,1 transaksi/detik pada traffic 100 hingga 32,6 transaksi/detik pada traffic 5000, sedangkan tanpa load balancing throughput bervariasi dari 72,1 transaksi/detik hingga 31,9 transaksi/detik pada traffic yang sama. Selain itu, delay pada sistem dengan load balancing IP hash menunjukkan penurunan dari 30 ms pada traffic 100 hingga 2 ms pada traffic 5000. Sebaliknya, tanpa load balancing, delay bervariasi dari 85 ms pada traffic 100 hingga mencapai 123 ms pada traffic 500, kemudian menurun menjadi 36 ms pada traffic 5000. Packet Loss Rate dengan load balancing IP hash tetap sangat rendah pada semua level traffic, berkisar antara 0,00% hingga 0,02%, sedangkan tanpa load balancing, Packet Loss Rate bervariasi dari 0,05% pada traffic 100 hingga 0,01% pada traffic 5000.

REFERENSI

Dani, R., & Suryawa, F. (2017). *Perancangan Dan Pengujian Load Balancing Dan Failover Menggunakan Nginx*. *3*(1), 43–50.

Denny, M., Pujangga, C., Hidayat, N., & Bakhtiar, F. A. (2021). *Implementasi Load Balancing Pada Cloud Computing Dengan Algoritme Improved Weighted Least Connection*. *5*(10), 4296–4303.

Fadila, A., Nasir, M., & Safriadi, S. (2023). Implementasi Sistem Load Balancing Web Server Pada Jaringan Public Cloud Computing Menggunakan Least Connection. *Journal Of Artificial Intelligence And Software Engineering (J-Aise)*, *3*(2), 50. Https://Doi.Org/10.30811/Jaise.V3i2.4578

Hasbi, M., & Saputra, N. R. (2021). *Analisis Quality Of Service (Qos) Jaringan Internet Kantor Pusat King Bukopin Dengan Menggunakan Wireshark*. *12*(1), 17–23.

Komaruddin, A. M., Sipitorini, D. M., & Rispian, P. (2019). *Load Balancing Dengan Metode Round Robin*. *5*(2), 47–50.

Magfa, B., Rahmat, R., & Nugraha, F. S. (2022). *Implementasi Load Balancing Metode Per Connection Classifier Dan Failover Recursive Menggunakan*. *6*(2), 284–289.

Rahmatulloh, A., & Msn, F. (2017). *Implementasi Load Balancing Web Server Menggunakan Haproxy Dan Sinkronisasi File Pada Sistem Informasi Akademik Universitas Siliwangi*. *02*, 241–248.

Saputra, E. P., Saryoko, A., Maulidah, M., Hidayati, N., & Dalis, S. (2023). Analisis Quality Of Service (Qos) Performa Jaringan Internet Wireless Lan Pt. Bhineka Swadaya Pertama. *Evolusi : Jurnal Sains Dan Manajemen*, *11*(1), 13–21. Https://Doi.Org/10.31294/Evolusi.V11i1.14955

Syaqia, A., & Asmunin. (2017). Implementasi Load Balancing Web Server Menggunakan Haproxy. *Jurnal Manajemen Informatika*, *08*(01), 11–19.

Warman, I., & Andrian, A. (2017). Analisis Kinerja Load Balancing Dua Line Koneksi Dengan Metode Nth (Studi Kasus: Laboratorium Teknik Informatika Institut Teknologi Padang) Indra. *Jurnal Teknoif Itp*, 1–7.