Peningkatan Skalabilitas Server Menggunakan HAProxy dengan Algoritma Load Balancing Round-Robin

Dany Purno Yuwono[1)*], Achmad Imam Agung[2)], I Gusti Putu Asto Buditjahjanto[3)], Lilik Anifah[4)]
[1)2)3)4)] S2 Pendidikan Teknologi dan Kejuruan, Fakultas Pascasarjana, Universitas Negeri Surabaya
*email: 24070895042@mhs.unesa.ac.id

DOI: https://doi.org/10.31603/komtika.v9i1.13405

Received: 19-04-2025, Revised: 14-05- 2025, Accepted: 17-05-2025

ABSTRACT

Server scalability is a crucial factor in maintaining the performance of web-based services. One effective method for improving scalability is load balancing, with HAProxy being a widely used solution for efficiently distributing traffic. This study aims to analyze how server performance improves by implementing HAProxy with the Round-Robin load balancing algorithm. The research methodology includes the installation of HAProxy, configuration of the Round-Robin algorithm, and performance testing under various workloads. The results demonstrate that implementing HAProxy can evenly distribute server loads and reduce response times. Therefore, HAProxy with the Round-Robin algorithm proves to be a powerful solution for enhancing server performance. As a result, the implementation of HAProxy with the Round-Robin algorithm is recommended as a reliable approach to improve server scalability, particularly for systems handling high traffic and requiring optimal load distribution.

Keywords: *Server Scalability, HAProxy, Load Balancing, Round-Robin.*

ABSTRAK

Skalabilitas server menjadi faktor penting dalam menjaga kinerja layanan berbasis web seiring meningkatnya jumlah pengguna. Salah satu pendekatan untuk meningkatkan skalabilitas adalah dengan menggunakan *load balancing*, di mana HAProxy menjadi salah satu pilihan yang banyak dimanfaatkan karena keandalannya. Penelitian ini bertujuan untuk menganalisis peningkatan kinerja server dengan implementasi HAProxy menggunakan algoritma *load balancing* Round-Robin. Metode penelitian yang diterapkan meliputi proses instalasi dan konfigurasi HAProxy, pengujian performa server dengan Apache JMeter, serta evaluasi distribusi beban kerja. Hasil penelitian ini mengindikasikan bahwa penggunaan HAProxy dengan algoritma Round-Robin efektif dalam menyebarkan beban secara merata dan secara signifikan mempercepat waktu respons server jika dibandingkan dengan sistem yang tidak menerapkan load balancing. Selain itu, kemampuan untuk menambah atau mengurangi jumlah server backend tanpa mempengaruhi kelancaran layanan yang ada menjadi salah satu keunggulan yang sangat menonjol. Oleh karena itu, penerapan HAProxy dengan algoritma Round-Robin direkomendasikan sebagai solusi efektif dalam meningkatkan skalabilitas server, terutama untuk sistem yang menangani lalu lintas tinggi dan membutuhkan distribusi beban yang optimal.

Keywords: Skalabilitas Server, HAProxy, Load Balancing, Round-Robin.

PENDAHULUAN

Perkembangan teknologi informasi (TI) di era digital saat ini telah memberikan dampak signifikan dalam berbagai sektor terutama dalam menunjang dan meningkatkan keefektifitasan dan keefisienan proses kerja yang ada dalam sebuah instansi [1]. Dalam era digital yang semakin berkembang, skalabilitas server menjadi isu global yang sangat penting. Dengan meningkatnya jumlah pengguna internet dan aplikasi berbasis web, kebutuhan akan sistem yang dapat menangani lalu lintas tinggi secara efisien semakin mendesak. Menurut

penelitian, penggunaan *load balancing* dapat secara signifikan meningkatkan kinerja server dan mengurangi waktu respons, yang sangat penting untuk menjaga kepuasan pengguna [2].

Di tingkat nasional, Indonesia mengalami pertumbuhan pesat dalam penggunaan layanan digital. Hal ini menuntut infrastruktur teknologi informasi yang lebih handal dan efisien. Implementasi solusi *load balancing*, seperti HAProxy, telah menjadi pilihan banyak perusahaan di Indonesia untuk meningkatkan keandalan dan skalabilitas server mereka. Penelitian menunjukkan bahwa HAProxy dapat mendistribusikan beban secara merata, yang pada gilirannya meningkatkan kinerja server [3].

Secara lokal, di Jakarta, banyak perusahaan teknologi dan startup yang mulai mengimplementasikan solusi *load balancing* untuk memastikan layanan mereka tetap optimal. Penelitian menunjukkan bahwa penggunaan HAProxy dengan algoritma *Round-Robin* efektif dalam mendistribusikan beban kerja secara merata, yang dapat meningkatkan waktu respons server dan mengurangi risiko downtime [4]. Untuk mendukung penerapan solusi ini, diperlukan perangkat lunak pendukung yang mampu mengukur kualitas layanan jaringan secara akurat, guna memastikan kinerja sistem tetap sesuai standar yang diharapkan [5].

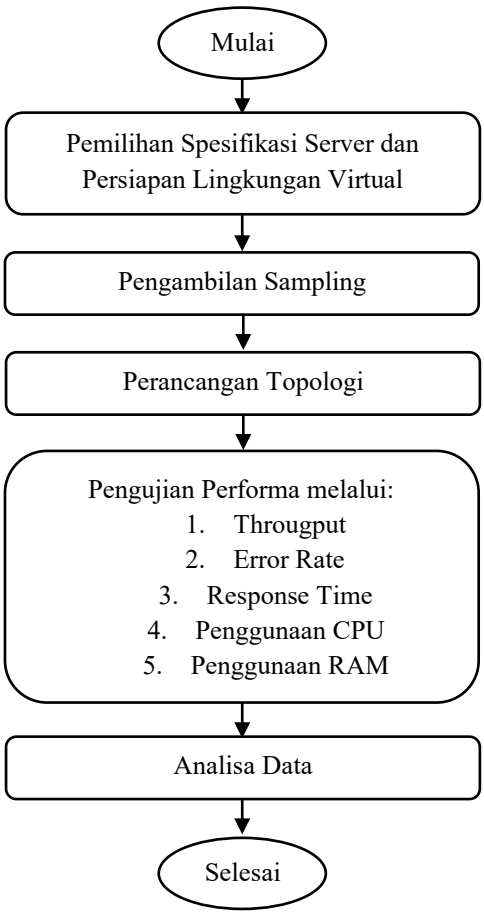
Berdasarkan perkembangan tersebut, penelitian ini menawarkan pendekatan baru dalam implementasi load balancing menggunakan HAProxy dengan algoritma *Round-Robin*. Kebaruan dari penelitian ini terletak pada tiga aspek utama, yaitu: pengujian dilakukan pada lingkungan virtualisasi berbasis VMware Workstation di laboratorium pendidikan, yang memberikan konteks unik pada skala sumber daya terbatas, analisis performa mencakup *multi-metrik* secara komprehensif, termasuk *throughput, error rate*, penggunaan CPU, dan RAM, dan yang terakhir dilakukan simulasi beban bertingkat menggunakan Apache JMeter untuk menguji ketahanan sistem terhadap peningkatan trafik secara gradual. Dengan pendekatan ini, penelitian diharapkan dapat memberikan kontribusi praktis dan akademis dalam optimalisasi server berbasis load balancing untuk kebutuhan pendidikan dan organisasi berskala menengah.

Penelitian ini bertujuan untuk menganalisis peningkatan kinerja server dengan implementasi HAProxy menggunakan algoritma *load balancing Round-Robin*. Dengan fokus pada instalasi, konfigurasi, dan pengujian performa menggunakan Apache JMeter, penelitian ini diharapkan dapat memberikan wawasan yang lebih dalam tentang efektivitas HAProxy dalam meningkatkan skalabilitas server.

METODE

Penelitian ini menggunakan pendekatan kuantitatif dengan desain eksperimen yang bertujuan untuk mengkaji implementasi sistem load balancing menggunakan HAProxy dalam lingkungan virtual. Kegiatan penelitian dilaksanakan di laboratorium komputer jurusan Teknik Komputer dan Jaringan (TKJ) SMKN 1 Turen, Kabupaten Malang, Jawa Timur, selama tiga hari, yaitu pada tanggal 24 hingga 26 Maret 2025. Populasi dalam penelitian ini terdiri atas sistem server dengan konfigurasi load balancing berbasis HAProxy, sedangkan sampel yang digunakan mencakup tiga server backend yang dijalankan secara virtual menggunakan VMware Workstation Pro. Pemilihan lingkungan laboratorium dan penggunaan simulasi virtual bertujuan untuk memastikan kontrol penuh terhadap variabel-variabel eksperimen yang dapat memengaruhi hasil penelitian.

Untuk memberikan pemahaman yang lebih sistematis terhadap tahapan-tahapan dalam pelaksanaan, penelitian ini menyajikan alur proses dalam bentuk diagram alir (*flowchart*) yang ditampilkan pada Gambar 1. Diagram ini menggambarkan langkah-langkah penting yang dilakukan selama penelitian, mulai dari pemilihan spesifikasi server dan persiapan lingkungan virtual, pengambilan sampling, perancangan topologi, hingga tahap pengujian dan analisis data. Representasi visual ini diharapkan dapat mempermudah dalam memahami prosedur penelitian secara menyeluruh serta mendukung transparansi metodologi yang digunakan.



Gambar 1. Diagram Alir Tahapan Penelitian

Sebagai langkah awal dalam pelaksanaan penelitian, pemilihan server dilakukan dengan mempertimbangkan keseragaman spesifikasi perangkat keras untuk memastikan validitas data dan mengurangi potensi bias dalam pengujian. Dalam tahapan ini, jumlah server yang digunakan terdiri dari:

1. Satu server load balancer (HAProxy)
2. Tiga server backend (Web Server 1, Web Server 2, dan Web Server 3). Server backend ini berfungsi sebagai penerima distribusi beban trafik dari server load balancer, dengan masing-masing melayani permintaan klien secara bergantian berdasarkan algoritma Round-Robin.

Dalam penelitian ini, metode sampling yang digunakan adalah *purposive sampling*, di mana pemilihan sampel dilakukan berdasarkan kriteria tertentu yang telah ditentukan, yaitu server dengan spesifikasi yang seragam dan dapat diakses melalui jaringan lokal di laboratorium.

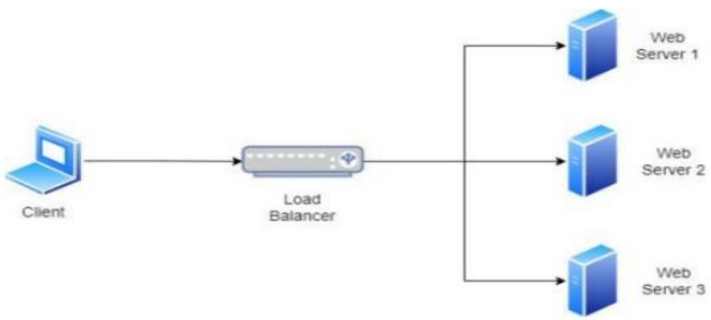
Adapun spesifikasi alat dan bahan yang digunakan dalam penelitian ini meliputi:

Tabel 1. Spesifikasi Alat dan Bahan

Kategori	Alat	Spesifikasi	Fungsi
Hardware	Server Backend	RAM 4 GB, CPU 2 Core, Storage 40 GB	Menyediakan layanan web untuk menerima beban dari load balancer
Hardware	Server Load Balancer	RAM 4 GB, CPU 2 Core, Storage 40 GB	Menjalankan HAProxy sebagai load balancer
Software	VMware Workstation Pro	Versi 17.5.0	Virtualisasi server backend dan load balancer
Software	HAProxy	Versi 2.2-stable	Load balancing trafik antara server backend
Software	Apache JMeter	Versi 5.6.2	Alat untuk simulasi trafik klien dan pengujian performa
Sistem Operasi	Debian Server	Versi 12.6.0	Sistem operasi server backend dan load balancer

Analisis data dilakukan dengan menggunakan pendekatan statistik deskriptif untuk menyajikan hasil pengujian secara jelas, serta menerapkan analisis komparatif untuk menilai kinerja server sebelum dan sesudah implementasi HAProxy. Penyajian data dilakukan dengan menggunakan format gambar, yang bertujuan untuk mempermudah pemahaman dan interpretasi hasil penelitian oleh pembaca. Dengan cara ini, informasi yang disajikan menjadi lebih jelas dan mudah diakses.

Perancangan topologi sistem ini melibatkan penggunaan *load balancer* yang berfungsi untuk mendistribusikan lalu lintas dari klien ke tiga server web yang tersedia. Dengan konfigurasi ini, *load balancer* akan menerima permintaan dari klien dan secara efisien mengarahkan permintaan tersebut ke salah satu dari tiga server, yaitu Web Server 1, Web Server 2, dan Web Server 3. Pendekatan ini bertujuan untuk meningkatkan ketersediaan dan kinerja sistem, serta memastikan bahwa beban kerja terdistribusi secara merata di antara server-server yang ada.



Gambar 2. Topologi HAProxy dan *Load Balancer* dengan 3 Web Server

Keabsahan hasil penelitian diverifikasi dengan cara melakukan pengulangan pengujian dalam kondisi yang serupa untuk memastikan konsistensi dan akurasi temuan. Selain itu,

analisis dilakukan untuk menilai keandalan data yang diperoleh, sehingga hasil penelitian dapat dianggap valid dan dapat diandalkan.

Uji coba sistem akan difokuskan pada lima aspek utama performa, yaitu:

1. Throughput
   Pengujian throughput bertujuan untuk mengukur kemampuan sistem dalam memproses sejumlah permintaan dalam satuan waktu, yang dinyatakan dalam satuan requests per second. Dalam uji ini, digunakan Apache JMeter sebagai alat bantu untuk mengirimkan permintaan secara paralel ke sistem, sehingga dapat mensimulasikan beban akses nyata dari pengguna. Data yang diperoleh berupa jumlah permintaan yang berhasil ditangani oleh sistem dalam interval waktu tertentu, memberikan gambaran mengenai kapasitas pemrosesan sistem secara menyeluruh. Melalui pengujian ini, dapat dievaluasi sejauh mana sistem mampu menangani beban tinggi tanpa mengalami penurunan performa.
2. Error Rate
   Aspek pengujian error rate difokuskan pada pengukuran tingkat kegagalan dalam penanganan permintaan oleh sistem. Tujuannya adalah untuk mengidentifikasi proporsi permintaan yang menghasilkan respons kesalahan, seperti kode status HTTP 4xx dan 5xx. Dalam pelaksanaannya, Apache JMeter kembali digunakan untuk mendeteksi dan mencatat jenis serta frekuensi kesalahan yang terjadi selama skenario pengujian. Analisis terhadap tren kesalahan pada berbagai tingkat beban akan memberikan wawasan mengenai keandalan dan stabilitas sistem dalam kondisi normal maupun ekstrem.
3. Response Time
   Pengujian response time dirancang untuk mengevaluasi waktu rata-rata yang diperlukan sistem dalam merespons suatu permintaan dari pengguna. Parameter ini menjadi indikator penting dalam menilai kecepatan dan efisiensi layanan yang diberikan sistem. Waktu respons dicatat sejak permintaan dikirim hingga respons diterima secara penuh. Pengujian ini dilakukan dengan dua skenario, yaitu sistem dengan dan tanpa penggunaan HAProxy, sehingga memungkinkan perbandingan performa dalam konteks penggunaan load balancer. Hasil dari pengujian ini berkontribusi pada pemahaman mengenai dampak arsitektur sistem terhadap kecepatan layanan.
4. Penggunaan CPU
   Efisiensi penggunaan sumber daya prosesor dievaluasi melalui pengujian penggunaan CPU. Fokus dari aspek ini adalah untuk mengetahui seberapa besar beban pemrosesan yang ditanggung oleh sistem pada berbagai tingkat permintaan. Pemantauan dilakukan terhadap konsumsi CPU di setiap server yang terlibat dalam pengujian. Persentase penggunaan CPU yang tercatat pada tiap skenario beban akan menjadi indikator efisiensi algoritma dan konfigurasi sistem dalam memanfaatkan sumber daya komputasi. Hasil ini juga penting dalam perencanaan kapasitas dan optimisasi kinerja.
5. Penggunaan RAM
   Aspek terakhir yang diuji adalah penggunaan memori (RAM), yang bertujuan untuk menilai kestabilan sistem dalam pengelolaan memori selama pengoperasian. Selama pengujian, penggunaan RAM dipantau secara kontinu untuk mendeteksi potensi kebocoran memori atau penggunaan yang tidak efisien. Perbandingan dilakukan antara sistem dengan dan tanpa HAProxy guna mengetahui pengaruh arsitektur terhadap konsumsi memori. Data

yang diperoleh memungkinkan analisis lebih lanjut terhadap efisiensi penggunaan memori dalam mendukung performa sistem secara keseluruhan.

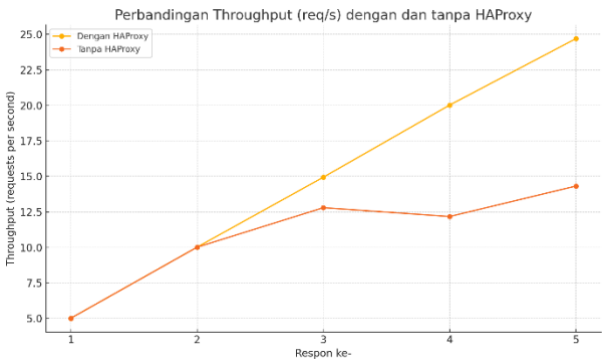
Dengan kelima analisa ini, hasil pengujian dapat mencerminkan performa sistem secara menyeluruh dan mendalam. Setiap metrik yang diuji memberikan kontribusi yang saling melengkapi untuk menggambarkan kondisi aktual dari sistem yang diuji, baik dari sisi efisiensi, stabilitas, maupun kapasitas pemrosesan. Throughput dan response time memberikan gambaran mengenai kecepatan dan kapasitas sistem dalam melayani permintaan, sedangkan error rate menunjukkan tingkat keandalan layanan. Di sisi lain, penggunaan CPU dan RAM mencerminkan efisiensi penggunaan sumber daya perangkat keras dalam berbagai skenario beban. Dengan demikian, pendekatan multi-metrik ini memungkinkan peneliti untuk menyusun rekomendasi yang berbasis data dalam upaya peningkatan kinerja dan skalabilitas sistem secara lebih akurat.

HASIL DAN PEMBAHASAN

Penelitian ini melibatkan tiga server backend yang dioperasikan dalam lingkungan laboratorium di TKJ SMKN 1 Turen, Kabupaten Malang. Setiap server memiliki spesifikasi yang serupa, yaitu dilengkapi dengan 4 GB RAM, 2 CPU, dan 40 GB storage. Server-server ini digunakan untuk menguji kinerja sistem dengan menerapkan HAProxy sebagai *load balancer*. Adapun hasil analisanya sebagai berikut:

1. Analisa *Throughput*

Pengujian performa menggunakan JMeter menunjukkan bahwa sistem dengan HAProxy, yang menerapkan mekanisme *load balance* menggunakan metode *round robin,* memiliki *throughput* yang lebih tinggi dan stabil dibandingkan sistem tanpa HAProxy. Hal ini dapat dilihat dalam visualisasi dalam Gambar 3. Dalam lima skenario uji, *throughput* pada sistem dengan HAProxy meningkat dari 5,01 hingga 24,69 permintaan per detik, sementara sistem tanpa HAProxy hanya mencapai kisaran 5,01 hingga 14,32 permintaan per detik, dengan kecenderungan stagnasi pada beban tinggi.



Gambar 3. Hasil Analisa *Throughput*

2. Analisa *Error rate*

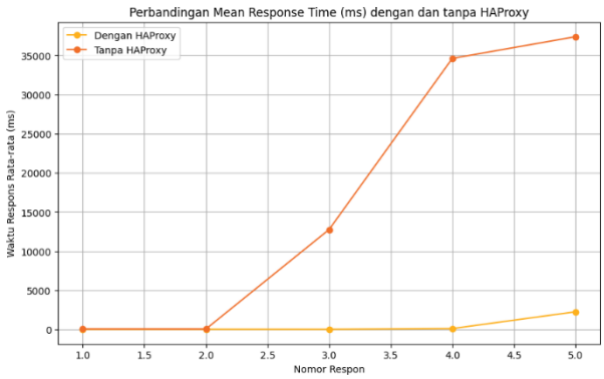
Dalam visualisasi Gambar 4 menunjukkan perbandingan *error rate* antara sistem dengan HAProxy dan tanpa HAProxy. Sistem yang menggunakan HAProxy dengan *load balance* menggunakan metode *round robin* menunjukkan kinerja yang sangat stabil, dengan *error rate* sebesar 0,00% pada seluruh skenario uji. Sebaliknya, sistem tanpa HAProxy mulai

menunjukkan kenaikan *error rate* yang signifikan setelah uji kedua, yaitu sebesar 5,78%, kemudian meningkat menjadi 10,61% pada uji keempat, dan sedikit menurun menjadi 9,64% pada uji kelima.

Gambar 4. Hasil Analisa *Error Rate*

3. Analisa *Response Time*

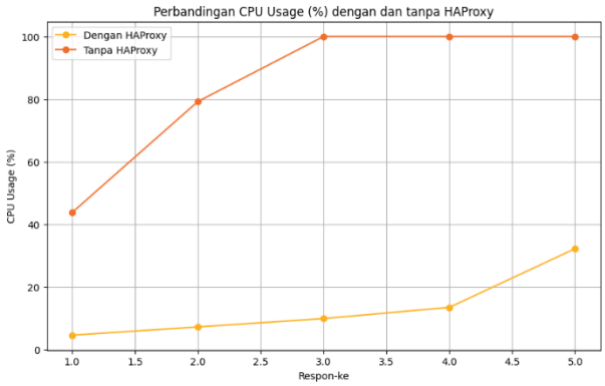
Dalam analisis performa system yang ditunjukkan oleh Gambar 5, penggunaan HAProxy sebagai *load balance* menggunakan metode *round robin* menunjukkan hasil yang signifikan dalam pengurangan waktu respons dibandingkan dengan pengujian tanpa HAProxy. Data dari pengujian JMeter menunjukkan bahwa rata-rata waktu respons untuk skenario dengan HAProxy meningkat secara bertahap dari 51.39 ms pada respon pertama hingga 2275.47 ms pada respon kelima, mencerminkan peningkatan beban dan kompleksitas permintaan. Sebaliknya, tanpa HAProxy, waktu respons rata-rata melonjak drastis dari 87.95 ms menjadi 37379.21 ms pada respon kelima, menandakan adanya masalah skalabilitas dan efisiensi dalam penanganan permintaan yang lebih tinggi.



Gambar 5. Hasil Analisa *Response Time*

4. Analisa CPU

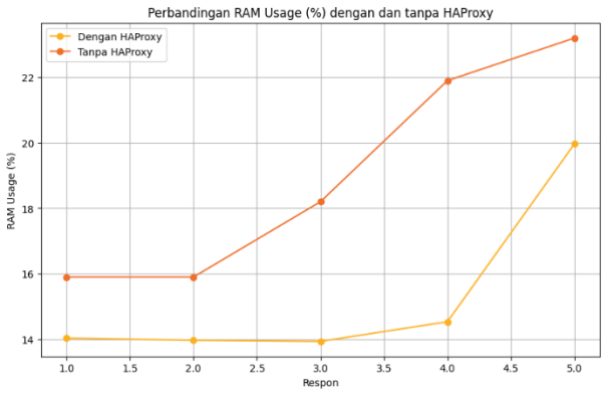
Pengujian performa dalam visualisasi Gambar 6 menunjukkan bahwa penggunaan HAProxy dengan metode load balancing secara signifikan mengurangi konsumsi CPU dibandingkan sistem tanpa HAProxy. Pada lima skenario pengujian, penggunaan CPU pada sistem dengan HAProxy mulai dari 4,53% pada pengujian pertama dan meningkat hingga 32,20% pada pengujian kelima. Sebaliknya, pada sistem tanpa HAProxy, penggunaan CPU melonjak drastis dari 43,70% pada pengujian pertama hingga mencapai 100% pada pengujian ketiga hingga kelima.



Gambar 6. Hasil Analisa CPU

5. Analisa RAM

Hasil pengamatan terhadap Gambar 7 konsumsi RAM menunjukkan bahwa pada pengujian pertama, sistem dengan HAProxy menggunakan RAM sebesar 14,03 MB, sedangkan sistem tanpa HAProxy menggunakan 15,90 MB. Pada pengujian kedua, nilai konsumsi RAM sistem dengan HAProxy berada di angka 13,97 MB, sementara tanpa HAProxy tetap sebesar 15,90 MB. Pengujian ketiga menunjukkan konsumsi RAM sebesar 13,93 MB untuk sistem dengan HAProxy dan 18,20 MB untuk sistem tanpa HAProxy. Selanjutnya, pada pengujian keempat, penggunaan RAM meningkat menjadi 14,53 MB untuk sistem dengan HAProxy dan 21,90 MB tanpa HAProxy. Terakhir, pada pengujian kelima, sistem dengan HAProxy menggunakan RAM sebesar 19,97 MB, sedangkan sistem tanpa HAProxy mencapai 23,20 MB.



Gambar 7 Hasil Analisa RAM

KESIMPULAN

Berdasarkan hasil analisis pengujian performa sistem, implementasi HAProxy sebagai *load balancer* menggunakan metode *round robin* memberikan peningkatan signifikan dalam efisiensi dan kestabilan sistem. Pada aspek *response time*, sistem dengan HAProxy menunjukkan waktu respons yang jauh lebih terkendali meskipun beban meningkat, dengan peningkatan bertahap dari 51,39 ms hingga 2275,47 ms. Sebaliknya, tanpa HAProxy, waktu respons melonjak tajam hingga 37379,21 ms, menandakan keterbatasan dalam menangani permintaan tinggi. Dari sisi *error rate*, sistem dengan HAProxy mencatat 0,00% kesalahan di seluruh skenario uji, sementara sistem tanpa HAProxy mengalami peningkatan *error* hingga

10,61% pada pengujian keempat, yang mencerminkan ketidakstabilan sistem tanpa load balancing. Dalam analisis *throughput*, sistem dengan HAProxy mampu mencapai hingga 24,69 permintaan per detik, jauh lebih baik dibandingkan sistem tanpa HAProxy yang hanya mencapai 14,32 permintaan per detik dan mengalami stagnasi saat beban tinggi.

Pada aspek efisiensi sumber daya, HAProxy menunjukkan dampak positif dalam menurunkan konsumsi CPU dan RAM. Sistem dengan HAProxy mencatat penggunaan CPU maksimum sebesar 32,20%, jauh lebih rendah dibandingkan sistem tanpa HAProxy yang mencapai 100% pada pengujian ketiga hingga kelima. Sementara itu, penggunaan RAM pada sistem dengan HAProxy tetap lebih rendah dan stabil, berkisar antara 13,93 MB hingga 19,97 MB, sedangkan sistem tanpa HAProxy mencapai 23,20 MB.

Secara umum, hasil eksperimen menunjukkan bahwa HAProxy merupakan solusi load balancing yang efektif untuk meningkatkan performa sistem, terutama dalam hal kestabilan respons, efisiensi sumber daya, dan kemampuan menangani permintaan tinggi. Oleh karena itu, penggunaan HAProxy sangat layak dipertimbangkan dalam lingkungan sistem yang menuntut performa tinggi, termasuk di sektor pendidikan atau institusi dengan volume permintaan layanan yang besar. Selain itu, uji coba terhadap algoritma *load balancing* lainnya seperti *least connection* atau *source hashing* juga direkomendasikan untuk memperluas pemahaman mengenai efisiensi dan efektivitas *load balancing* dalam skenario yang berbeda.

DAFTAR PUSTAKA

[1] T. Titah and T. Sutabri, "Evaluasi Tata Kelola Teknologi Informasi pada Puskesmas Sukarami Menggunaka Framework Cobit 5 Domain Evaluate, Direct, and Monitor (EDM)," *J. Komtika (Komputasi dan Inform.*, vol. 8, no. 2, pp. 162–170, Nov. 2024, doi: 10.31603/komtika.v8i2.12648.

[2] F. Apriliansyah, I. Fitri, and A. Iskandar, "Implementasi Load Balancing Pada Web Server Menggunakan Nginx," *J. Teknol. Dan Manaj. Inform.*, vol. 6, no. 1, pp. 18–26, 2020, [Online]. Available: https://doi.org/10.26905/jtmi.v6i1.3792

[3] D. K. Hakim, J. K. Riyanto, and A. Fauzan, "Pengujian Algoritma Load Balancing pada Virtualisasi Server," *Sainteks*, vol. 16, no. 1, pp. 33–41, 2020, [Online]. Available: https://doi.org/10.30595/sainteks.v16i1.7015

[4] H. Wijaya, I. Abdurrohim, J. Tugiyono, and R. J. Rumandan, "Implementasi Metode Load Balancing Untuk Optimalisasi Performa Server Pada Jaringan Internet," *J. Inf. Syst. Res.*, vol. 5, no. 1, pp. 252–260, 2023, [Online]. Available: https://doi.org/10.47065/josh.v5i1.4386

[5] V. A. Saputro and Y. Z. Arief, "Studi Komparasi Software berbasis GUI dan CLI Terhadap Pengukuran Kualitas Throughput Jaringan Nirkabel IEEE 802.11ac," *J. Komtika (Komputasi dan Inform.*, vol. 8, no. 1, pp. 14–21, May 2024, doi: 10.31603/komtika.v8i1.11218.

[6] S. D. Riskiono and D. Pasha, "Analisis Perbandingan Server Load Balancing dengan Haproxy & Nginx dalam Mendukung Kinerja Server E-Learning," *J. Telekomun. dan Komput.*, vol. 10, no. 3, p. 135, 2020, [Online]. Available: https://doi.org/10.22441/incomtech.v10i3.8751

[7] R. Dani and F. Suryawan, "Perancangan dan Pengujian Load Balancing dan Failover Menggunakan Nginx," *Khazanah Inform. J. Ilmu Komput. dan Inform.*, vol. 3, no. 1, pp. 43–50, 2017, [Online]. Available: https://doi.org/10.23917/khif.v3i1.2939

[8] H. Triangga, I. Faisal, and I. Lubis, "Analisis Perbandingan Algoritma Static Round-Robin dengan Least-Connection Terhadap Efisiensi Load Balancing pada Load Balancer Haproxy," *InfoTekJar (Jurnal Nas. Inform. dan Teknol. Jaringan)*, vol. 4, no. 1, pp. 70–75, 2019, [Online]. Available: https://doi.org/10.30743/infotekjar.v4i1.1688

[9] A. Hanafiah and R. Wandri, "Implementasi Load Balancing Dengan Algoritma Penjadwalan Weighted Round Robin Dalam Mengatasi Beban Webserver," *IT J. Res. Dev.*, vol. 5, no. 2, pp. 226–233, 2021, [Online]. Available: https://doi.org/10.25299/itjrd.2021.vol5(2).5795

[10] E. P. Cynthia, I. Iskandar, and A. A. Sipayung, "Rancang Bangun Server HAproxy Load Balancing Master to Master MySQL (Replication) Berbasis Cloud Computing," *Algoritm. J. Ilmu Komput. dan Inform.*, vol. 4, no. 1, p. 45, 2020, [Online]. Available: https://doi.org/10.30829/algoritma.v4i1.7275

[11] D. K. Hakim, D. Y. Yulianto, and A. Fauzan, "Pengujian Algoritma Load Balancing pada Web Server Menggunakan NGINX," *JRST (Jurnal Ris. Sains dan Teknol.*, vol. 3, no. 2, p. 85, 2019, [Online]. Available: https://doi.org/10.30595/jrst.v3i2.5165

[12] A. M. Komaruddin, D. M. Sipitorini, and P. Rispian, "Load Balancing dengan Metode Round Robin Untuk Pembagian Beban Kerja Web Server," *Siliwangi*, vol. 5, no. 2, pp. 47–50, 2019, [Online]. Available: https://jurnal.unsil.ac.id/index.php/jssainstek/article/view/1184

[13] H. G. Tani and C. El Amrani, "Smarter Round Robin Scheduling Algorithm for Cloud Computing and Big Data," *J. Data Min. Digit. Humanit.*, vol. Special Is, pp. 1–8, 2018, [Online]. Available: https://doi.org/10.46298/jdmdh.3104

[14] V. Mohammadian, N. J. Navimipour, M. Hosseinzadeh, and A. Darwesh, "Fault-Tolerant Load Balancing in Cloud Computing: A Systematic Literature Review," *IEEE Access*, vol. 10, pp. 12714–12731, 2022, [Online]. Available: https://doi.org/10.1109/ACCESS.2021.3139730

[15] T. S. Hendrana and I. M. Suartana, "Penerapan Container Load Balancing untuk Manajemen Trafik pada Learning Manajemen System," *J. Informatics Comput. Sci.*, vol. 4, pp. 169–182, 2022, [Online]. Available: https://doi.org/10.26740/jinacs.v4n02.p169-182

[16] H. Halimatusyadiah, Y. Afrianto, and B. A. Prakosa, "Implementasi Sistem Load Balancing Pada Web Server Berbasis Raspberry Pi Dengan Metode Long Short-Term Memory," *J. Pendidik. dan Teknol. Indones.*, vol. 4, no. 11, pp. 345–358, 2024, [Online]. Available: https://doi.org/10.52436/1.jpti.477

[17] H. Nasser and T. Witono, "Analisis Algoritma Round Robin, Least Connection, dan Ratio pada Load Balancing Menggunakan OPNET Modeler," *J. Inform.*, vol. 12, no. 1, pp. 25–32, 2016, [Online]. Available: https://doi.org/10.21460/inf.2016.121.455

[18] I. G. Primanata, N. Putra Sastra, and D. M. Wiharta, "Load Balancing untuk Perbandingan Kinerja Virtualisasi Server VMware dan XEN Server," *J. SPEKTRUM*, vol. 5, no. 1, p. 32, 2018, [Online]. Available: https://doi.org/10.24843/SPEKTRUM.2018.v05.i01.p05

[19] A. Setiawan, M. R. A. Suhendi, and E. Arlitasari, "Pembuatan Load Balancer untuk Server Moodle di Sekolah Vokasi IPB University," *J. Inform. Teknol. dan Sains*, vol. 5, no. 2, pp. 251–257, 2023, [Online]. Available: https://doi.org/10.51401/jinteks.v5i2.2674

[20] 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, [Online]. Available: https://doi.org/10.32832/kreatif.v8i2.3503

[21] A. R. Sofyan and S. D. Y. Kusuma, "Implementasi Load Balancing Web Server menggunakan Haproxy pada Virtual Server Direktorat SMK Kemendikbudristek," *J. Pendidik. Tambusai*, vol. 6, pp. 9669–9682, 2022, [Online]. Available: https://jptam.org/index.php/jptam/article/view/3954