Jurnal Informatika Universitas Pamulang **ISSN: 2541-1004**
Penerbit: Fakultas Ilmu Komputer Universitas Pamulang **e-ISSN: 2622-4615**
Vol. 9, No. 2, Juni 2024 (38-45) **https://doi.org/10.32493/informatika.v9i2.42662**

Load Balancing untuk Lalu Lintas Tinggi pada Lingkungan Cloud Menggunakan Metode Round Robin

Dodi Siregar[1], Afrilian Ariangga[2*], Sarudin[3], Herlina Harahap[4], Risko Liza[5]
[1]Teknik Informatika, Fakultas Teknik dan Komputer, Universitas Harapan Medan, Medan-Indonesia, 20151
Email: [1]dodisiregar@unhar.ac.id

[2,3,4,5] Teknik Informatika, Fakultas Teknik dan Komputer, Universitas Harapan Medan, Medan-Indonesia, 20151
Email: [2]afrilian.ariangga@gmail.com, [3]sarudin@unhar.ac.id, [4]herlinaharahap411@gmail.com, [5]risko.liza@gmail.com
*Corresponding author

Submitted Date: June 11th, 2024 Reviewed Date: June 20th, 2024
Revised Date: July 23th, 2024 Accepted Date: July 30th, 2024

Abstract

The rapid development of technology has changed many aspects of the data communication process, especially in cloud computing. Cloud computing or commonly known as cloud computing is now widely used. With the increase in cloud usage, there is a significant increase in traffic and causes a load imbalance between servers in the cloud environment. To overcome load imbalance on cloud servers, this research proposes the use of load balancing with a round robin method. Load balancing divides traffic equally between servers to prevent overload and system failure. The round robin algorithm ensures fair distribution by sending requests alternately to available servers. Testing shows a significant reduction in error rates when using load balancing. As many as 30 thousand users per second, the error dropped to 42.37%, 50 thousand users per second caused an error of 66.31%, and 100 thousand users per second resulted in an error of 63.08%. This indicates increased system scalability and ability to effectively handle surges in user requests. Overall, implementing load balancing using the round robin method is able to improve system performance in handling high traffic in a cloud environment, maintain stability, and minimize the risk of failure.

Keywords: cloud computing, google cloud, load balancing, round robin, overload.

Abstrak

Perkembangan teknologi yang begitu pesat telah mengubah banyak aspek dalam proses komunikasi data khususnya dalam komputasi awan. cloud computing atau biasa dikenal dengan komputasi awan sekarang sudah banyak digunakan, dengan peningkatan penggunaan cloud, terjadi peningkatan lalu lintas yang signifikan dan menyebabkan ketidakseimbangan beban (load imbalance) di antara server-server yang ada dalam lingkungan cloud. Untuk mengatasi ketidakseimbangan beban di server-server cloud, penelitian ini mengusulkan penggunaan load balancing dengan metode round robin. Load balancing membagi lalu lintas secara seimbang di antara server untuk mencegah overload dan kegagalan sistem. Algoritma round robin memastikan distribusi yang adil dengan mengirim permintaan secara bergantian ke server-server yang tersedia. Uji coba menunjukkan penurunan signifikan dalam tingkat kesalahan saat menggunakan load balancing. Sebanyak 30 ribu pengguna per detik, error turun menjadi 42.37%, 50 ribu pengguna per detik menyebabkan error sebesar 66.31%, dan 100 ribu pengguna per detik menghasilkan error 63.08%. Ini menandakan peningkatan skalabilitas sistem dan kemampuan menangani lonjakan permintaan pengguna secara efektif. Secara keseluruhan, implementasi load balancing dengan metode round robin mampu meningkatkan kinerja sistem dalam mengatasi lalu lintas tinggi pada lingkungan cloud, menjaga stabilitas, dan meminimalkan risiko kegagalan.

Kata kunci: cloud computing, google cloud, load balancing, round robin, overload.

This work is licensed under a Creative Commons Attribution-NonCommercial 4.0 International (CC BY-NC 4.0) License
Copyright © 2024 Dodi Siregar, Afrilian Ariangga, Sarudin, Herlina Harahap, Risko Liza

1. Pendahuluan

Perkembangan teknologi yang begitu pesat telah mengubah banyak aspek dalam proses komunikasi data. Perkembangan teknologi tersebut membuat proses komunikasi data berkembang untuk semakin memudahkan manusia dalam memperoleh informasi. Hal ini dapat terlihat dari meningkatnya permintaan dan penggunaan layanan *cloud* dalam beberapa tahun terakhir. *Cloud computing* telah menjadi pendekatan yang populer untuk menyediakan sumber daya IT yang fleksibel dan skalabel bagi perusahaan dan organisasi (Santoso, 2023).

Teknologi *cloud computing* atau biasa dikenal dengan komputasi awan sekarang sudah banyak digunakan dan bukan hal yang asing lagi. Teknologi *cloud computing* dapat lebih menghemat pengeluaran dibandingkan harus membangun sendiri infrastruktur jaringan untuk jangka pendek. Namun, dengan peningkatan penggunaan *cloud*, terjadi peningkatan lalu lintas pada infrastruktur *cloud*. Lalu lintas ini dapat terdiri dari permintaan *web*, penggunaan aplikasi, transaksi basis data, atau operasi lain yang memerlukan sumber daya komputasi. Peningkatan lalu lintas pada lingkungan *cloud* dapat menyebabkan beberapa masalah yang perlu diatasi. Salah satu masalah yang umum adalah ketidakseimbangan beban (*load imbalance*) di antara server-server yang ada dalam lingkungan *cloud* (Sabila & Rahayu, 2024).

Server adalah suatu sistem komputer yang memiliki layanan khusus berupa penyimpanan data. Server akan menyimpan beragam jenis dokumen dan menyediakan informasi umtuk pengguna atau pengunjungnya (Nugraha et al., 2023). Server menyediakan berbagai macam layanan, beberapa diantara layanan tersebut adalah otentikasi, keamanan, web, berkas dan data. Dalam sebuah jaringan terdapat minimal satu buah server yang berfungsi untuk menyediakan layanan untuk klien (Hapsari et al., 2020).

Menurut Institusi Nasional Standar dan Teknologi, definisi *cloud computing* adalah model untuk meningkatkan kenyamanan, memberikan *on-demand access* ke jaringan terminal sumber daya *cloud computing* bersama yang dapat dikonfigurasi yaitu jaringan, server, penyimpanan, aplikasi dan layanan yang diberikan yang dapat ditetapkan dengan cepat dan dirilis dengan upaya manajemen atau interaksi penyedia layanan yang minimal (’Abidah et al., 2020; Rakha Allam et al., n.d.).

Load balancing adalah teknik untuk mendistribusikan beban *traffic* pada dua atau lebih jalur koneksi secara seimbang, agar *traffic* dapat berjalan optimal, memaksimalkan *throughput*, memperkecil waktu tanggap dan menghindari *overload* pada salah satu jalur koneksi (Ardianto et al., 2018; Hakim et al., 2019; Jamil & Hamid, n.d.). Load balancing juga mendistribusikan beban kerja secara merata di dua atau lebih komputer, link jaringan, CPU, hard drive, atau sumber daya lainnya, untuk mendapatkan pemanfaatan sumber daya yang optimal (Linggom Parlindungan Panjaitan et al., n.d.; Rexa et al., 2019). Pada dasarnya teknologi load balancing ini untuk memanfaatkan server secara efektif, sehingga jaringan komputer dapat bekerja dengan baik tanpa adanya suatu kendala (Wibowo et al., 2018).

Round robin merupakan salah satu algoritma penjadwalan proses dalam sistem operasi. *Round robin* dirancang untuk membagi waktu setiap proses pada porsi yang sama dan dalam urutan melingkar, menjalankan semua proses tanpa prioritas dikenal juga sebagai eksekutif siklik. Penjadwalan *round robin* mudah diterapkan, dan bebas starvation. Penjadwalan *round robin* juga dapat diterapkan untuk masalah penjadwalan lainnya, seperti penjadwalan paket data dalam jaringan komputer. *Round robin* dirancang untuk sistem time sharing (Cynthia et al., n.d.; Komaruddin et al., 2019). *Round Robin* adalah penjadwalan proses menerapkan strategi *preemptive*, bukan *di-preempt* oleh proses lain, tapi terutama oleh penjadwal berdasarkan jatah waktu pemroses yang disebut kwanta (*quantum*) (Fadli et al., 2020).

Penggunaan nginx sebagai load balancing untuk pengatur lalu lintas dan pembagian beban kerja server, ketika salah satu server mendapatkan beban trafik yang tinggi hingga *down* maka akan dialihkan ke server lainnya untuk mendapatkan akses situs yang diinginkan tanpa adanya gangguan *overload* pada server (Bustomi et al., 2019).

Dalam penelitian ini, dibangun server dalam lingkungan *cloud* menggunakan teknologi *load balancing* dan metode *round robin* untuk mengalokasikan lalu lintas yang tinggi, seperti permintaan web, penggunaan aplikasi, dan transaksi basis data, secara bergantian ke server-server yang tersedia. Tujuan utamanya adalah

This work is licensed under a Creative Commons Attribution-NonCommercial 4.0 International (CC BY-NC 4.0) License
Copyright © 2024 Dodi Siregar, Afrilian Ariangga, Sarudin, Herlina Harahap, Risko Liza

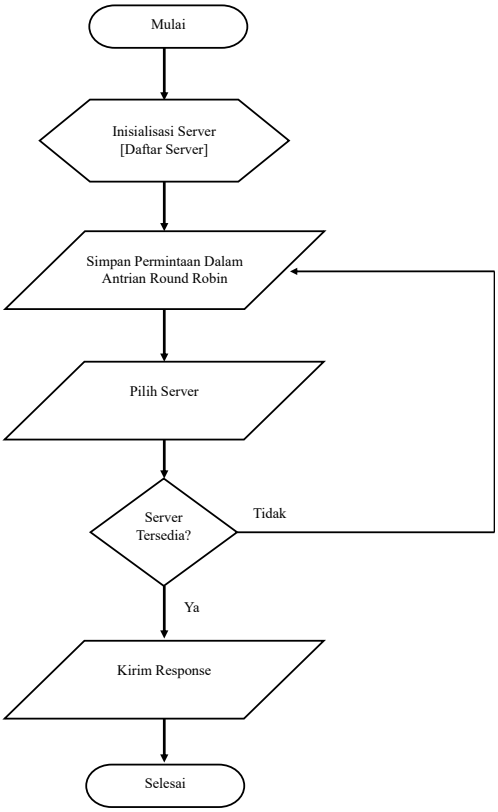
ISSN: 2541-1004
e-ISSN: 2622-4615
https://doi.org/10.32493/informatika.v9i2.42662

untuk meningkatkan kinerja sistem, mencegah kelebihan beban pada server, serta mengurangi risiko server mengalami *overload*.

2. Metode Penelitian

2.1. Flowchart Load Balancing

Flowchart dapat diartikan sebagai langkah langkah penyelesaian masalah yang di tuliskan dalam suatu simbol-simbol tertentu. Diagram alir ini akan menunjukan alur di dalam program secara logika (Khesya, 2021; Syamsiah, 2019).



Gambar 1. *Flowchart Load Balancing*

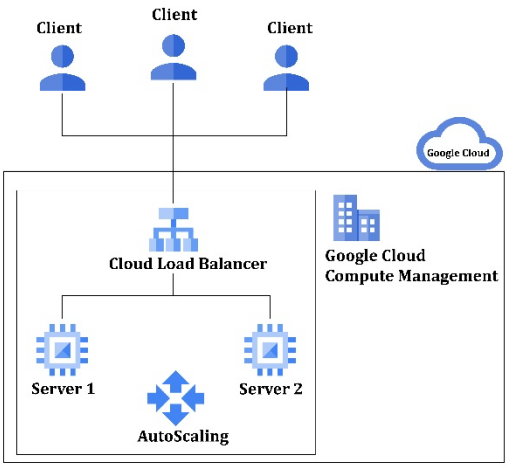
Gambar 1 menggambarkan cara kerja *load balancing* dimana rinciannya adalah sebagai berikut:

1. Mulai, Proses mulai sebuah sistem *load balancing.*
2. Inisialisasi server list yang tersedia, *load balancing* akan membuat nama dan jumlah server yang tersedia.
3. Simpan permintaan dalam antrian *round robin, load balancing* akan menyimpan semua permintaan yang mengarah ke server ke dalam antrian menggunakan metode *round robin.*
4. Pilih Server, *load balancing* akan memilih server yang tersedia secara berurutan sesuai dengan algoritma *round robin.*
5. Server tersedia, *load balancing* akan melakukan pengecekan ke server,jika server tersedia maka permintaan akan diteruskan ke server jika server tidak tersedia maka permintaan akan kembali masuk kedalam antrian.
6. Kirim *response*, *load balancing* akan mengirimkan *response* ke *client* sesuai dari server yang dipilih.
7. Selesai, Proses selesai sebuah sistem *load balancing.*

2.2. Topologi Load Balancing

Topologi jaringan adalah salah satu aturan bagaimana menghubungkan komputer (*node*) satu sama lain secara fisik dan pola hubungan antara komponen-komponen yang berkomunikasi melalui media atau peralatan jaringan, seperti server, workstation, hub/switch, dan pemasangan kabel (media transmisi data) (Prayoga & Kurniawan, 2020; Widodo et al., 2018).

Topologi akan menggambarkan susunan atau arsitektur dari perangkat dan koneksi yang saling terhubung membentuk jaringan, pada gambar 2 adalah topologi yang digunakan untuk load balancing.



Gambar 2. Topologi *Load Balancing*

Pada gambar 2 Load balancer berperan sebagai pintu masuk untuk menerima permintaan dari pengguna dan mendistribusikan lalu lintas ke server secara merata ke server 1 dan server 2, dan autoscaling berperan penting untuk menambah

This work is licensed under a Creative Commons Attribution-NonCommercial 4.0 International (CC BY-NC 4.0) License
Copyright © 2024 Dodi Siregar, Afrilian Ariangga, Sarudin, Herlina Harahap, Risko Liza

dan mengurangi sumber daya server secara dinamis sesuai dengan beban kerja yang dibutuhkan,setiap permintaan akan dikirim ke server melalui load balancer sesuai antrian server menggunakan metode round robin.

2.3. Metode Round Robin

Adapun ketentuan algoritma Round robin adalah sebagai berikut: (Pranowo Kuswandono & Saepuloh, 2021; Sylvia & Kurniawan, 2019)

1. Jika proses sebelumnya belum selesai maka proses menjadi runnable atau pemrosesan dialihkan ke proses lain atau berikutnya.
2. Jika waktu quantum belum habis dan proses masih berjalan (selesainya operasi I/O), maka proses akan di blocked dan pemrosesan akan dilaihkan ke proses berikutnya.
3. Jika waktu quantum belum habis dan proses telah selesai maka proses diakhiri dan melanjutkan ke proses lainnya.

Untuk mengetahui bagaimana cara menghitung penjadwalan Round robin. Masing-masing proses memiliki waktu datang (*Arrival Time*) dan waktu proses (*Burst Time*) dengan masing-masing memiliki nilai. Waktu datang merupakan waktu dimana proses mulai mengantri, sedangkan waktu proses (*Burst Time*) merupakan kemungkinan waktu yang digunakan untuk memproses antrian dalam satu proses. Nilai pada *quantum time* merupakan waktu untuk pemprosesan yang sebenarnya.

Tabel 1 merupakan contoh proses *round robin* dimana nilai *quantum time* = 4 detik.

Tabel 1. Proses *round robin*

Nama Proses	Waktu Datang	Burst Time
P1	1	3
P2	4	5
P3	6	7
P4	8	10
P5	10	12

Langkah pertama untuk menyelesaikan algoritma Round Robin adalah dengan memasukkan proses yang berurutan, kemudian kurangi Burst Time dengan nilai quantum seperti pada tabel 2.

Tabel 2. Sisa Burst Time

Nama Proses	Sisa Burst Time		
	R0	R1	R2
P1	0	0	0
P2	1	0	0
P3	3	0	0
P4	6	2	0
P5	8	4	0

Pada tabel 2 sisa Burst Time didapat dari nilai Burst Time dikurangi dengan nilai Quantum dan Jika Nilai Burst Time ≤ Quantum Time Maka Sisa Burst Time = 0 atau Tidak Ada Sisa.
Langkah Untuk Mencari Nilai R0 atau putaran pertama:

P1: Burst Time – Quantum Time
3-4 = -1
Sisa Burst Time R0 = 0

P2: Burst Time-Quantum Time
5-4=1
Sisa Burst Time R0 = 1

P3: Burst Time-Quantum Time
7-4=3
Sisa Burst Time R0 = 3

P4: Burst Time-Quantum Time
10-4=6
Sisa Burst Time R0 = 6

P5: Burst Time-Quantum Time
12-4=8
Sisa Burst Time R0 = 8

Proses yang memiliki sisa, kemudian dimasukkan kembali kedalam antrian sesuai dengan urutan prosesnya Untuk Mencari Nilai R1 atau putaran kedua.

P2: Sisa Burstime R0 – Quantum Time
1-4= -4
Sisa Burstime R1 = 0

P3: Sisa Burstime R0 – Quantum Time
3-4= -1
Sisa Burstime R1 = 0

P4: Sisa Burstime R0 – Quantum Time
6-4= 2
Sisa Burstime R1 = 2

P5: Sisa Burstime R0 – Quantum Time
8-4= 4
Sisa Burstime R1 = 4

This work is licensed under a Creative Commons Attribution-NonCommercial 4.0 International (CC BY-NC 4.0) License
Copyright © 2024 Dodi Siregar, Afrilian Ariangga, Sarudin, Herlina Harahap, Risko Liza

ISSN: 2541-1004
e-ISSN: 2622-4615
https://doi.org/10.32493/informatika.v9i2.42662

Proses yang memiliki sisa, kemudian dimasukkan kembali ke dalam antrian sesuai dengan urutan prosesnya untuk mencari nilai R2 atau putaran ketiga.

P4: Sisa Burstime R1 – Quantum Time
2-4= -2
Sisa Burstime R2 = 0

P5: Sisa Burstime R1 – Quantum Time
4-4= 4
Sisa Burstime R2 = 0

Selanjutnya buatlah *gantt chart* dan masukan jumlah proses sebanyak jumlah *Quantum Time* pada contoh ini *Quantum Time* = 4 detik, Jika *Burst Time* < *Quantum Time* maka jumlah proses hanya sebesar *Burst Time*, buat proses berurutan sesuai dengan tabel 2. Tabel 3 merupakan *gantt chart* yang telah dibuat.

Tabel 3. *Gantt chart* proses *round robin*

Proses	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37
P1																																						
P2																																					
P3																																					
P4																																					
P5																																					

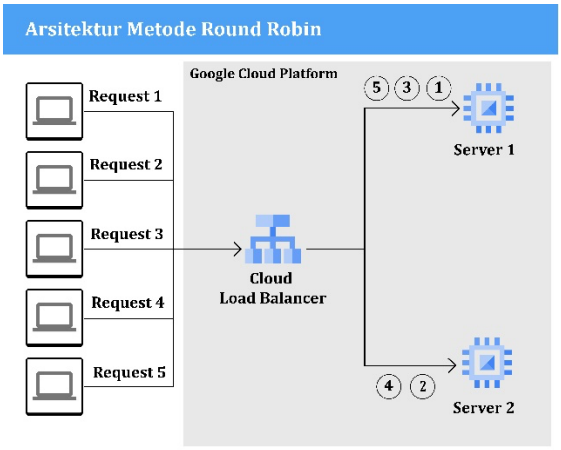
Dari tabel 3 didapatkan waktu mulai dan waktu selesai seperti yang dirincikan pada tabel 4.

Tabel 4. waktu mulai dan waktu selesai proses

Nama Proses	Waktu Mulai	Waktu Selesai
P1	1	3
P2	4	20
P3	8	23
P4	12	33
P5	16	37

2.4. Arsitektur Round Robin

Gambar 3 adalah arsitektur cara kerja algoritma round robin.



Gambar 3 Arsitektur Metode *Round Robin*

Pada gambar 3 terdapat lima *client* yang melakukan permintaan (*request)* kepada server, *load balancer* menggunakan metode *round robin* akan memilih server yang tersedia sesuai antrian untuk mengirim permintaan seperti yang dijelaskan berikut ini:

1. *Client* 1 mengirim *request, load balancer* akan mengirim *request* ke server 1 karena antrian pertama pada server 1
2. *Client* 2 mengirim *request, load balancer* akan mengirim *request* ke server 2 karena antrian pada server 1 sudah terisi dan server 2 masi kosong
3. *Client* 3 mengirim *request, load balancer* akan mengirim *request* ke server 1 karena antrian selanjutnya adalah server 1
4. *Client* 4 mengirim *request, load balancer* akan mengirim *request* ke server 2 karena antrian selanjutnya adalah server 2
5. *Client* 5 mengirim *request, load balancer* akan mengirim *request* ke server 1 karena antrian selanjutnya adalah server 1
6. Begitu seterusnya jika ada *request, load balancer* akan mengirim *request* ke server sesuai antrian.

3. Hasil dan Pembahasan

Pada tahap ini dilakukan pengujian sistem yang sudah dibangun untuk menampilkan hasil yang sudah dirancang pada tahap sebelumnya maupun sebelum dirancang yaitu implementasi *load balancing* untuk lalu lintas tinggi pada lingkungan *cloud* menggunakan metode *round robin* yang diuji menggunakan Apache JMeter 5.5.

This work is licensed under a Creative Commons Attribution-NonCommercial 4.0 International (CC BY-NC 4.0) License
Copyright © 2024 Dodi Siregar, Afrilian Ariangga, Sarudin, Herlina Harahap, Risko Liza

Tabel 5. pengujian sebelum menggunakan *load balancing*

Banyak Pengguna	*Error*	*Throughput*	*Received*	*Sent*	*Average*
30 Ribu	62.28%	204.9Kb/s	1091.98Kb/s	8.91Kb/s	5456.6 B
50 Ribu	75.39%	670.2Kb/s	2891.26Kb/s	19.01Kb/s	4417.7 B
100 Ribu	73.59%	334.2Kb/s	1477.04Kb/s	10.17Kb/s	4525.4 B

Hasil pengujian sebelum menerapkan load balancing pada tabel 5 menunjukkan beberapa temuan penting yang perlu dipertimbangkan. Pertama, terdapat peningkatan dalam *throughput* (jumlah data yang berhasil ditransfer dalam satu periode waktu) saat jumlah pengguna meningkat. Namun, hal ini diimbangi oleh peningkatan yang proporsional dalam jumlah *error* yang terjadi. Pada uji coba dengan 30 ribu pengguna, terlihat bahwa throughput mencapai 204.9Kb/s, yang menunjukkan kemampuan sistem untuk menangani volume data yang cukup besar. Namun, tingkat *error* yang mencapai 62.28% menunjukkan adanya kelemahan dalam stabilitas atau kapasitas sistem yang belum dioptimalkan.

Ketika jumlah pengguna meningkat menjadi 50 ribu dan 100 ribu, *throughput* meningkat secara pasti menjadi 670.2Kb/s dan 334.2Kb/s berturut-turut. Meskipun demikian, tingkat *error* juga meningkat menjadi 75.39% dan 73.59% secara berurutan, menandakan bahwa peningkatan jumlah pengguna menyebabkan penurunan kualitas layanan secara keseluruhan.

Tabel 6. pengujian setelah menggunakan *load balancing*

Banyak Pengguna	*Error*	*Throughput*	*Received*	*Sent*	*Average*
30 Ribu	42.37%	226.5Kb/s	1527.13Kb/s	14.66Kb/s	6904.01 B
50 Ribu	66.31%	331.3Kb/s	1650.57Kb/s	12.53Kb/s	5102.4 B
100 Ribu	63.08%	331.1Kb/s	1733.28Kb/s	13.81Kb/s	5329.2 B

Setelah menerapkan *load balancing*, hasil pengujian pada tabel 6 menunjukkan peningkatan dalam kinerja sistem, Terutama, terlihat bahwa tingkat *error* telah mengalami penurunan yang berarti dari pengujian sebelumnya, menunjukkan peningkatan dalam stabilitas dan kualitas layanan. Pada uji coba dengan 30 ribu pengguna, terlihat bahwa *throughput* meningkat menjadi 226.5Kb/s, sementara tingkat *error* menurun menjadi 42.37%. Hal ini menunjukkan bahwa implementasi l*oad balancing* telah berhasil meningkatkan efisiensi pengelolaan beban kerja, sehingga sistem mampu menangani *volume* data yang lebih besar dengan tingkat keberhasilan yang lebih tinggi.

Ketika jumlah pengguna meningkat menjadi 50 ribu dan 100 ribu, *throughput* juga terus meningkat, mencapai 331.3Kb/s dan 331.1Kb/s berturut-turut. Meskipun tingkat *error* masih cukup tinggi, yaitu 66.31% dan 63.08%, peningkatan *throughput* yang menunjukkan bahwa sistem telah lebih efektif dalam menangani beban kerja yang lebih besar. Hasil pengujian setelah menggunakan *load balancing* menunjukkan peningkatan yang berarti dalam kinerja sistem, dengan penurunan tingkat *error* dan peningkatan *throughput* yang konsisten. Meskipun demikian, masih perlu dilakukan optimisasi lebih lanjut untuk mengatasi tingkat *error* yang masih cukup tinggi pada kapasitas beban kerja yang lebih besar.

4. Kesimpulan

Membangun dan menerapkan *load balancing* dalam lingkungan *cloud* untuk lalu lintas tinggi menggunakan *virtual machine* yang disediakan oleh *google cloud* untuk membangun server, implementasi *load balancing* dengan metode *round robin* dapat meningkatkan kinerja sistem dan mendistribusikan beban kerja server secara merata dalam mengatasi lalu lintas tinggi pada lingkungan *cloud*. Metode *round robin* berhasil mengurangi waktu *respons* sistem dan mempercepat waktu pemrosesan permintaan pengguna. Uji coba dengan mengirim permintaan

This work is licensed under a Creative Commons Attribution-NonCommercial 4.0 International (CC BY-NC 4.0) License
Copyright © 2024 Dodi Siregar, Afrilian Ariangga, Sarudin, Herlina Harahap, Risko Liza

ke server menggunakan *load balancing* dengan 30 ribu pengguna per detik menghasilkan *error* sebanyak 42.37%, 50 ribu pengguna per detik menghasilkan *error* sebanyak 66.31% dan 100 ribu pengguna per detik menghasilkan *error* sebanyak 63.08% lebih sedikit daripada sebelum menggunakan *load balancing* dengan 30 ribu pengguna per detik menghasilkan *error* sebanyak 62.28%, 50 ribu pengguna per detik menghasilkan *error* 75.39% sebanyak dan 100 ribu pengguna per detik menghasilkan *error* sebanyak 73.59%.

Referensi

Abidah, I. N., Hamdani, M. A., & Amrozi, Y. (2020). Implementasi Sistem Basis Data Cloud Computing pada Sektor Pendidikan. *KELUWIH: Jurnal Sains Dan Teknologi*, *1*(2), 77–84. https://doi.org/10.24123/saintek.v1i2.2868

Ardianto, F., Alfaresi, B., & Darmadi, A. (2018). RANCANG BANGUN LOAD BALANCING DUA INTERNET SERVICE PROVIDER (ISP) BERBASIS MIKROTIK. *Jurnal Surya Energy*, *3*(1).

Bustomi, Z., Syahiruddin, M., Afandi, M. I., Fahmi, K., Holle, H., Informatika, J. T., Islam, U., Maulana, N., Malang, M. I., Gajayana, J., 50, N., Lowokwaru, K., Malang, K., & Timur, J. (2019). *Load Balancing Web Server Menggunakan Nginx pada Lingkungan Virtual*. *xx, No.xx*.

Cynthia, E. P., Iskandar, I., Sipayung, A. A., Informatika, J. T., Sains, F., Teknologi, D., & Riau, K. (n.d.). *Rancang Bangun Server HAproxy Load Balancing Master to Master MySQL (Replication) Berbasis Cloud Computing*.

Fadli, S., Ashari, M., & Imtihan, K. (2020). *SISTEM PENJADWALAN EVENT ORGANIZER DENGAN METODEROUND ROBIN (RR)*.

Hakim, D. K., Yulianto, D. Y., & Fauzan, A. (2019). Pengujian Algoritma Load Balancing pada Web Server Menggunakan NGINX. *JRST (Jurnal Riset Sains Dan Teknologi)*, *3*(2), 85. https://doi.org/10.30595/jrst.v3i2.5165

Hapsari, M. P., Prasetijo, A. B., & Fauzi, A. (2020). Analisa Kinerja pada Standalone Server dan Clustering Server Teknologi RAC (Real Application Clustering) dengan Algoritma DNS (Domain Name System) Round Robin Berbasis Oracle Linux 6.4 di Lingkungan Virtual. In *Tahun* (Vol. 10, Issue 2).

Jamil, M., & Hamid, M. (n.d.). *ANALISIS PERBANDINGAN KINERJA LOAD BALANCING MENGGUNAKAN METODE PCC DAN NTH*.

Khesya, N. (2021). *MENGENAL FLOWCHART DAN PSEUDOCODE DALAM ALGORITMA DAN PEMROGRAMAN*.

Komaruddin, A. M., Sipitorini, D. M., & Rispian, P. (2019). LOAD BALANCING DENGAN METODE ROUND ROBIN UNTUK PEMBAGIAN BEBAN KERJA WEB SERVER. *Jurnal Siliwangi*, *5*(2).

Linggom Parlindungan Panjaitan, S., Abdurohman, M., & Musthofa Jadied, E. (n.d.). *Analisis Perbandingan Performansi Load Balancing Menggunakan Algoritma Weighted Round Robin dengan Weighted Least Connection pada Software Defined Network*.

Nugraha, R. A., Ruhwan, & Hartono, R. (2023). *Merancang IP Failover & Replikasi Database Menggunakan Heartbeat Pada Komputer Server (STUDI KASUS : SMK PERIWATAS TASIKMALAYA) INFORMASI ARTIKEL A B S T R A K* (Vol. 5, Issue 2). https://e-journal.unper.ac.id/index.php/informatics

Pranowo Kuswandono, D., & Saepuloh, A. (2021). *ANALISIS TROUGHPUT DAN WAKTU RESPON WEB SERVER MENGGUNAKAN LOAD BALANCE DENGAN ALGORITMA ROUND ROBIN*. *11*.

This work is licensed under a Creative Commons Attribution-NonCommercial 4.0 International (CC BY-NC 4.0) License
Copyright © 2024 Dodi Siregar, Afrilian Ariangga, Sarudin, Herlina Harahap, Risko Liza

Prayoga, I., & Kurniawan, W. (2020). PENGEMBANGAN SERVER VOIP BERBASIS ASTERISK DAN SHOREWAL MENGGUNAKAN METODE NETWORK DEVELOPMENT LIFE CYLCE. In *Jurnal Satya Informatika* (Vol. 5, Issue 2).

Rakha Allam, D., Haryo Sulaksono, D., Nurina Prabiantissa, C., Gusti Eka Yuliastuti, dan, Teknik informatika, J., & Teknologi Adhi Tama Surabaya, I. (n.d.). *Sistem Pembelajaran Interaktif Menggunakan Cloud Computing Berbasis Client-Server di Jurusan Teknik Informatika ITATS*.

Rexa, M., Bella, M., Data, M., & Yahya, W. (2019). *Implementasi Load Balancing Server Web Berbasis Docker Swarm Berdasarkan Penggunaan Sumber Daya Memory Host* (Vol. 3, Issue 4). http://j-ptiik.ub.ac.id

Sabila, K., & Rahayu, S. (2024). *Peningkatan Efisiensi Penggunaan Sumber Daya Jaringan Melalui Teknik Load Balancing*. *4*(3). https://doi.org/https://doi.org/10.55606/cemerlang.v4i3.2989

Santoso, J. T. (2023). *Komputasi Awan (Cloud Computing)*. Yayasan Prima Agus Teknik Bekerja sama dengan Universitas Sains & Teknologi Komputer (Universitas STEKOM). https://digilib.stekom.ac.id/assets/dokumen/ebook/feb_Ds2GPNzqUQcumI-dzULM-KqX7d8-crBWgDah9j-GkA_TotwF-3_TTg_1673258004.pdf

Syamsiah. (2019). *PERANCANGAN FLOWCHART DAN PSEUDOCODE PEMBELAJARAN MENGENAL ANGKA DENGAN ANIMASI UNTUK ANAK PAUD RAMBUTAN*.

Sylvia, A., & Kurniawan, R. (2019). *Seminar Nasional Hasil Penelitian dan Pengabdian 2019 IBI DARMAJAYA Bandar Lampung*.

Wibowo, A., Virgono, A., & Latuconstina, R. (2018). *LOAD BALANCING PADA CLOUD COMPUTING MENGGUNAKAN METODE LEAST CONNECTION LOAD BALANCING ON CLOUD COMPUTING USING LEAST CONNECTION METHOD*.

Widodo, C., Yana, M., & Agung, H. (2018). *IMPLEMENTASI TOPOLOGI HYBRID UNTUK PENGOPTIMALAN APLIKASI EDMS PADA PROJECT OFFICE PT PHE ONWJ*.

This work is licensed under a Creative Commons Attribution-NonCommercial 4.0 International (CC BY-NC 4.0) License
Copyright © 2024 Dodi Siregar, Afrilian Ariangga, Sarudin, Herlina Harahap, Risko Liza