JURNAL FASILKOM
Volume 13 No. 3 | Desember 2023: 424-430

P-ISSN : 2089-3353
E-ISSN : 2808-9162

Evaluasi *Load Balancing*: Studi Komparatif *Least-Connection* dan *Round-Robin* dalam Konteks *Cloud Computing*

Deny Ariestiandy[1], Lilik Suhery[2], Jusmawati[3], Yanuardi[4]
[1]Program Studi Komputerisasi Akuntansi, AMIK Citra Buana Indonesia
[2]Program Studi Teknik Komputer, Sekolah Tinggi Teknologi Payakumbuh
[3]Program Studi Sistem Informasi, Fakultas Ilmu Komputer, Universitas Yapis Papua
[4]Program Studi Teknik Informatika, Fakultas Teknik, Universitas Muhammadiyah Tangerang
[1]denyariestiandycbi@gmail.com *, [2]liliksuhery@gmail.com, [3]juzmawati.nr@gmail.com, [4]yanuardi99@gmail.com

Abstract

Cloud computing technology requires an efficient load balancing method to distribute the workload evenly across all nodes in the network. This research aims to analyze the comparative effectiveness between load balancing algorithms, namely Round-Robin and Least-Connection. Testing was carried out over several different connection levels, from 1000/600 to 5000/1000 to assess the average response time results measured in milliseconds (ms). The empirical data collected indicates that there is a linear increase in response time in line with the increase in the number of connections in both algorithms. Although both show almost similar performance in the initial phase, the average Round-Robin response time on 1000/600 connections is 2.8 ms, while the Least-Connection reaches 2.9 ms. The Least-Connection algorithm shows consistent superiority at higher connections. On 5000/1000 connections, Round-Robin was recorded to have a response time of 3,275 ms, while Least-Connection had an average response time of 3.34 ms, which indicates that Least-Connection has better scalability in managing high-level connections. These findings suggest that for systems that expect a relatively stable and predictable number of connections, Round-Robin is recommended. However, for systems with dynamic and unpredictable workloads, Least-Connection has better capabilities.

Keywords: Cloud computing, Load Balancing, Least-Connection, Round-Robin, Response Time

Abstrak

Teknologi *cloud computing* membutuhkan metode *load balancing* yang efisien untuk mendistribusikan beban kerja secara merata ke seluruh node dalam jaringan. Penelitian ini bertujuan untuk menganalisis perbandingan efektivitas antara algoritma *load balancing* yaitu *Round-Robin* dan *Least-Connection*. Pengujian dilakukan melalui beberapa tingkat koneksi yang berbeda, dari 1000/600 hingga 5000/1000 untuk menilai hasil rata-rata *response time* yang diukur dalam millisecond (ms). Data empiris yang terkumpul mengindikasikan bahwa terjadi peningkatan linear *response time* sejalan dengan peningkatan jumlah koneksi pada kedua algoritma. Meskipun keduanya menunjukkan kinerja yang hampir serupa pada fase awal, dengan rata-rata *response time Round-Robin* pada 1000/600 koneksi adalah 2.8 ms, sedangkan *Least-Connection* mencapai 2.9 ms. Algoritma *Least-Connection* menunjukkan keunggulan yang konsisten di koneksi yang lebih tinggi. Pada 5000/1000 koneksi, *Round-Robin* tercatat memiliki *response time* 3.275 ms, sedangkan *Least-Connection* memiliki rata-rata *response time* 3.34 ms, yang mengindikasikan bahwa *Least-Connection* memiliki skalabilitas yang lebih baik dalam mengelola koneksi bertingkat tinggi. Temuan ini menyarankan bahwa untuk sistem yang mengharapkan jumlah koneksi yang relatif stabil dan dapat diprediksi maka direkomendasikan menggunakan *Round-Robin*. Namun, untuk sistem dengan beban kerja yang dinamis dan tidak terprediksi, maka *Least-Connection* memiliki kemampuan yang lebih baik.

Kata kunci: Cloud Computing, Load Balancing, Least-Connection, Round-Robin, Response Time

1. Pendahuluan

Dalam era digital saat ini, teknologi *cloud computing* telah menjadi bagian integral dari infrastruktur IT di banyak organisasi. Namun, tantangan utama dalam penerapan teknologi ini adalah manajemen beban kerja yang efisien untuk memastikan layanan yang konsisten dan andal [1]. Kekurangan dalam manajemen beban kerja dapat menyebabkan penurunan kinerja sistem dan peningkatan waktu tunggu bagi pengguna [2]. Untuk mengatasi masalah ini, berbagai teknik *load balancing* telah dikembangkan. Teknik *load balancing* bertujuan untuk mendistribusikan beban kerja secara merata di antara node dalam jaringan sehingga tidak ada satu node pun yang terbebani secara berlebihan [3]. Terdapat dua metode *load balancing* yang populer diantaranya adalah *Least-Connection* dan *Round-Robin* [4]. Metode *Least-Connection* mengarahkan permintaan baru ke server dengan jumlah koneksi terkecil, sedangkan *Round-Robin* mendistribusikan permintaan secara bergiliran ke setiap server dalam grup [5], [6]. Meskipun kedua metode ini telah banyak digunakan, belum ada studi komprehensif tentang performa mereka dalam konteks *cloud computing*. *Cloud computing* adalah agregasi dari sumber daya komputer dan jaringan yang menyediakan struktur tarif yang didasarkan pada penggunaan manajemen penyimpanan serta dukungan untuk aplikasi-aplikasi virtual yang beragam [7]. *Cloud computing* memudahkan dalam penggunaan sumber daya komputasi melalui jaringan internet, seperti

P-ISSN : 2089-3353
E-ISSN : 2808-9162

penyimpanan data, server, database, jaringan, perangkat lunak, dan sejumlah layanan lainnya [8]. Teknologi ini menawarkan kemudahan dalam penyesuaian kapasitas layanan sesuai dengan permintaan, sambil memperhatikan aspek-aspek ekonomis dan faktor lain yang relevan [9], [10]. Sejumlah penelitian telah memanfaatkan *cloud computing* sebagai alternatif dalam menyediakan infrastruktur yang diperlukan untuk mendukung layanan berbasis web.

Beberapa penelitian terkait telah dilakukan sebelumnya. Penelitian mengenai paradigma komputasi awan adalah keseimbangan beban yang tepat atas sumber daya yang tersedia [11]. Pada penelitian ini menghasilkan mengenai mekanisme keseimbangan beban telah dikembangkan untuk mengoptimalkan pemanfaatan sumber daya yang tersedia dan meningkatkan kinerja secara keseluruhan. Penelitian lainnya mengenai metode *Least-Connection* yang lebih efisien dibandingkan *Round-Robin* dalam lingkungan dengan trafik tinggi tetapi tidak merata [12]. Pada penelitian ini mengungkapkan bahwa algoritma penyeimbangan beban dinamis diimplementasikan dan cocok di lingkungan *cloud* yang heterogen komputasinya. Selanjutnya, penelitian mengenai membandingkan beberapa strategi *load balancing* termasuk *Least-Connection* dan *Round-Robin* dalam konteks virtualization dan menemukan bahwa pilihan strategi sangat bergantung pada karakteristik beban kerja spesifik mereka [13]. Dimana untuk koneksi 1000 dan 1200 *round robin* terlihat lebih baik. Namun pada pengujian koneksi dari 1400, 1600 dan 1800 terlihat *least connection* lebih unggul. Terakhir, penelitian yang membahas pentingnya *load balancing* untuk efisiensi sistem *cloud* serta optimasi performa sistem melalui pendekatan dinamis [14].

Namun demikian, masih ada kesenjangan pengetahuan tentang bagaimana perbandingan antara *Least-Connection* dan *Round-Robin* khususnya pada situasi-situasi tertentu seperti distribusi beban kerja yang merata atau tidak merata. Penelitian ini bertujuan untuk mengisi kesenjangan tersebut dengan melakukan evaluasi komprehensif dari dua teknik tersebut dalam konteks *cloud computing* menggunakan simulasi berbasis menggunakan mesin virtual. Penelitian ini diharapkan menghasilkan temuan yang akan memberikan wawasan penting bagi pengembang dan administrator jaringan tentang pemilihan strategi *load balancing* yang paling sesuai dengan kebutuhan mereka.

2. Metode Penelitian

2.1. Tahapan Penelitian

Penelitian ini dirancang untuk melakukan evaluasi komprehensif terhadap dua teknik *load balancing*, yaitu *Least-Connection* dan *Round-Robin*, dalam konteks *cloud computing*. Tahapan penelitian ini melibatkan beberapa langkah penting seperti yang diperlihatkan pada Gambar 1 berikut:

Gambar 1. Alur Penelitian

Penelitian ini diawali dengan membentuk sebuah lingkungan simulasi berbasis *cloud*. Lingkungan simulatif ini dirancang dengan menggunakan sejumlah server virtual yang terkoneksi dan beroperasi bersama-sama untuk menangani permintaan dari klien. Dalam tahap pembentukan ini, memanfaatkan sebuah perangkat lunak simulasi jaringan yang bersifat open source dan sudah terbukti kehandalannya dalam berbagai penelitian terdahulu. Selanjutnya, mengimplementasikan teknik *load balancing* menggunakan *Least-Connection* dan *Round-Robin* dalam sistem yang telah dipersiapkan. Selanjutnya merancang dan mengintegrasikan algoritma yang spesifik untuk masing-masing metode dengan tujuan untuk mendistribusikan beban kerja secara efisien di antara server-server yang ada dalam jaringan. Dalam fase pengujian beban kerja, dihasilkan berbagai skenario yang dirancang untuk menguji efektivitas dari kedua teknik *load balancing*. Skenario-skenario ini mencakup variasi dari beban kerja yang merata, dimana setiap permintaan memiliki kebutuhan sumber daya yang sama hingga beban kerja yang tidak merata dengan permintaan yang memiliki kebutuhan sumber daya yang signifikan perbedaannya. Setelah proses pengujian, dilakukan pengumpulan dan analisis data yang berkaitan dengan *response time* yang didapatkan dari sejumlah koneksi yang diberikan. Analisis ini dilakukan untuk mengevaluasi dan membandingkan performa dari kedua metode *load balancing*. Pada akhir penelitian, dilakukan perbandingan komprehensif terhadap hasil yang diperoleh dari kedua teknik tersebut. Penelitian ini memfokuskan pada efisiensi dalam distribusi beban kerja serta pengaruhnya terhadap latensi respon dan throughput untuk menarik kesimpulan akhir yang relevan dengan tujuan penelitian.

2.2. Skenario Pengujian

Pada penelitian ini akan melakukan analisis dan evaluasi terhadap kinerja dari pendekatan *load balancing*. Load balancing merupakan suatu metode yang digunakan untuk mendistribusikan lalu lintas jaringan secara merata ke berbagai server guna meningkatkan kinerja dan ketersediaan sistem. Dalam rangkaian penelitian ini, akan dikembangkan skenario pengujian yang dirancang untuk mengukur dan mengevaluasi *response time* dalam lingkungan yang

P-ISSN : 2089-3353
E-ISSN : 2808-9162

telah diimplementasikan dengan teknik *load balancing*. Penelitian ini menetapkan lima skenario pengujian yang masing-masing diidentifikasi berdasarkan tingkat koneksi yang berbeda. Setiap skenario diatur dengan rasio koneksi khusus untuk menguji kapasitas dan efektivitas dari algoritma *load balancing* dalam menangani jumlah permintaan yang bervariasi.

Skenario pertama diawali dengan 1000 koneksi aktif dan 600 koneksi pasif untuk menguji kondisi pada beban kerja dasar. Melangkah ke skenario kedua, tingkat koneksi ditingkatkan menjadi 2000 koneksi aktif dan 700 koneksi pasif, yang memberikan insight tentang performa algoritma di bawah beban yang lebih berat. Skenario ketiga meningkatkan jumlah ini menjadi 3000 koneksi aktif dan 800 koneksi pasif, sementara skenario keempat beroperasi pada 4000 koneksi aktif dan 900 koneksi pasif. Skenario terakhir, yang menandai kondisi dengan beban paling berat, memuat 5000 koneksi aktif dan 1000 koneksi pasif. Tabel 1 merupakan skenario dari pengujian koneksi.

Tabel 1. Spesifikasi Mesin Virtual

No	Skenario	Koneksi Aktif	Koneksi pasif
1	Skenario 1	1000	600
2	Skenario 2	2000	700
3	Skenario 3	3000	800
4	Skenario 4	4000	900
5	Skenario 5	5000	1000

Dalam setiap skenario, *response time* dicatat dan dianalisis. Dari data yang dihasilkan, diharapkan dapat mengidentifikasi bagaimana masing-masing algoritma menyesuaikan diri dengan peningkatan jumlah permintaan dan bagaimana ini mempengaruhi waktu yang dibutuhkan untuk memproses setiap permintaan tersebut. Pendekatan ini memungkinkan untuk secara kuantitatif menilai performa algoritma *Round-Robin* dan *Least-Connection* dalam menghadapi skala beban kerja yang meningkat secara progresif.

2.3. Rancangan Sistem Server

Untuk tujuan penelitian ini, yaitu merancang sistem server yang mencerminkan lingkungan *cloud computing*. Berikut adalah rincian dari rancangan sistem server:

1) Arsitektur Server

Menggunakan arsitektur berbasis cluster, di mana sejumlah server virtual saling terhubung dan bekerja bersama untuk memproses permintaan dari klien. Setiap server dalam cluster ini memiliki sumber daya (misalnya, CPU, memori) yang sama dan mampu memproses permintaan secara independen.

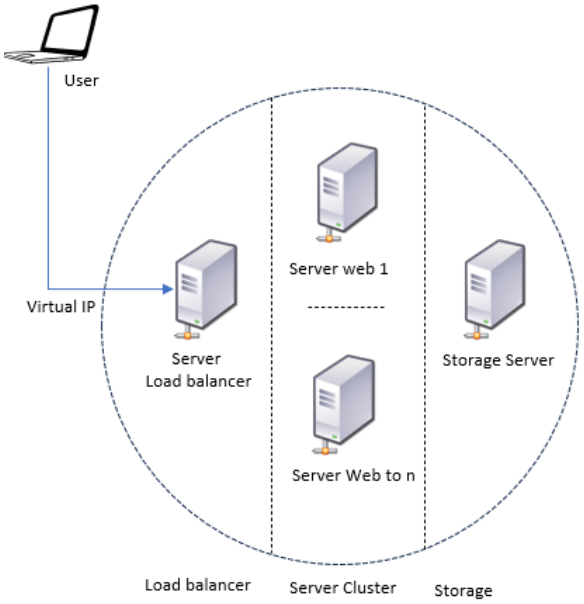
2) Teknik *Load balancing*

Load balancing merupakan metode yang digunakan untuk mendistribusikan beban kerja secara merata di antara beberapa sumber daya komputasi. Tujuan utamanya adalah untuk mencegah *overload* pada satu titik atau server tertentu, sehingga meningkatkan kinerja, kehandalan, dan efisiensi sistem secara keseluruhan. Dua teknik *load balancing* yaitu *Least-Connection* dan *Round-Robin* diimplementasikan dalam sistem ini. Dalam metode *Least-Connection*, permintaan baru dialihkan ke server dengan jumlah koneksi aktif terkecil. Di sisi lain, dalam metode *Round-Robin*, permintaan didistribusikan secara bergiliran ke setiap server dalam grup.

3) Manajemen Sumber Daya

Untuk memastikan bahwa setiap server tidak terbebani secara berlebihan serta melakukan implementasi mekanisme manajemen sumber daya yang dinamis. Mekanisme ini memantau penggunaan sumber daya oleh setiap server dan melakukan penyesuaian jika diperlukan (misalnya, menambah atau mengurangi jumlah sumber daya).

Rancangan sistem ini dirancang untuk memberi kita kontrol penuh atas lingkungan pengujian serta fleksibilitas untuk menyesuaikan parameter seperti jumlah total permintaan dan karakteristik beban kerja sesuai dengan skenario pengujian yang berbeda.



Gambar 2. Arsitektur Server Load Balancer

Dalam gambar 2, elemen utama yang dipresentasikan adalah *server load balancer*, yang memiliki peran kunci dalam mengalokasikan beban koneksi ke berbagai server web. Server ini diidentifikasi dengan alamat IP unik yang memfasilitasi interkoneksi dengan server-server aplikasi serta perangkat klien. Elemen kedua yang tercakup adalah kelompok server web, yang dalam lingkup studi ini terdiri dari dua entitas berfungsi sebagai server aplikasi web. Setiap server di antara kedua entitas ini mengoperasikan program aplikasi web yang dapat dijangkau dan digunakan oleh pengguna [15]. Elemen ketiga adalah komponen penyimpanan data atau database, yang untuk keperluan penelitian ini diintegrasikan langsung dalam masing-masing server web, bukan sebagai entitas yang berdiri

P-ISSN : 2089-3353
E-ISSN : 2808-9162

sendiri. Pengujian terhadap infrastruktur sistem ini dilaksanakan di dalam sebuah setup mesin virtual, dengan detail spesifikasi yang telah diuraikan dalam Tabel 2.

Tabel 2. Spesifikasi Mesin Virtual

Komponen	CPU	RAM	SSD	Sistem Operasi
Mesin Virtual Load Balancer	AMD Ryzen 5 (1 CPU) - 2,0 GHz	1 GB	20 GB	Ubuntu Server
Mesin Virtual Server 1	AMD Ryzen 5 (1 CPU) - 2,0 GHz	1 GB	20 GB	Ubuntu Server
Mesin Virtual Server 2	AMD Ryzen (1 CPU) - 2,0 GHz	1 GB	20 GB	Ubuntu Server
Mesin Virtual Klien	AMD Ryzen 5 (1 CPU) - 2,0 GHz	1 GB	20 GB	Ubuntu Server

Tabel 1 menampilkan rincian spesifikasi perangkat keras dan konfigurasi mesin virtual yang diterapkan dalam uji coba sistem ini, meliputi data mengenai prosesor atau CPU, RAM, SSD dan jenis sistem operasi yang dipasang pada tiap elemen. Sedangkan untuk menggambarkan alokasi alamat IP yang diberikan kepada tiap peranti yang termasuk dalam jaringan ini ditunjukkan pada Tabel 3.

Tabel 3. Nama Komponen dan Alamat IP

Komponen	Alamat IP
Server *Load balancing*	192.168.109.184
Server Web_1	192.168.109.179
Server Web_2	192.168.109.178
User	192.168.109.182

Tabel 2 menampilkan detail alokasi alamat IP untuk tiap perangkat yang berada dalam jaringan. Tercatat bahwa alamat IP yang diberikan untuk server penyeimbang beban adalah 192.168.109.184. Server ini memiliki fungsi esensial dalam mendistribusikan permintaan layanan yang masuk dari pengguna ke berbagai server web yang tersedia. Ketika pengguna mencoba mengakses layanan lewat server web, mereka akan diarahkan secara otomatis ke server penyeimbang beban ini. Server penyeimbang beban bertugas untuk mengalokasikan permintaan tersebut ke server web yang sesuai di latar belakang. Dalam sistem ini, pengguna tidak perlu mengetahui server web mana yang akan menangani permintaannya; hal ini sepenuhnya diatur oleh server penyeimbang beban. Server ini secara efisien mendistribusikan beban layanan kepada server web yang tersedia sesuai dengan skema alokasi yang telah ditetapkan sebelumnya.

2.4. Algoritma *Least-Connection*

Least-Connection adalah teknik *load balancing* yang populer dan efektif, terutama dalam lingkungan di mana beban kerja tidak merata atau permintaan memiliki kebutuhan sumber daya yang berbeda [13]. Dalam metode ini, distribusi beban kerja dilakukan dengan mempertimbangkan jumlah koneksi aktif pada setiap server. Server yang memiliki jumlah koneksi aktif paling sedikit akan menerima permintaan selanjutnya, sehingga memastikan bahwa beban kerja terdistribusi secara efisien. Algoritma *Least-Connection* memungkinkan sistem *load balancing* untuk secara dinamis menyesuaikan distribusi beban kerja berdasarkan kondisi aktual setiap server. Dengan demikian, algoritma ini membantu menghindari situasi di mana satu atau beberapa server mungkin mengalami beban kerja yang berlebihan sementara yang lain masih memiliki kapasitas yang belum terpakai.

Teknik ini berfungsi dengan mengarahkan permintaan baru ke server dengan jumlah koneksi aktif terkecil. Dengan kata lain, server yang sedang mengerjakan tugas paling sedikit akan mendapatkan permintaan baru. Pendekatan ini berdasarkan asumsi bahwa server dengan jumlah koneksi aktif terkecil kemungkinan besar memiliki sumber daya paling banyak yang tersedia untuk memproses permintaan baru. Oleh karena itu, teknik ini membantu memastikan bahwa tidak ada satu server pun yang terbebani secara berlebihan dan bahwa beban kerja didistribusikan seefisien mungkin di antara semua server [16]. Namun, metode *Least-Connection* juga memiliki beberapa batasan. Misalnya, metode ini mungkin tidak efektif jika semua permintaan memiliki kebutuhan sumber daya serupa karena dalam kasus seperti itu, pendekatan *Round-Robin* atau lainnya mungkin lebih baik [17].

2.5. Algoritma *Round-Robin*

Round-Robin adalah salah satu teknik *load balancing* yang paling sederhana dan banyak digunakan [18]. Teknik ini berfungsi dengan mendistribusikan permintaan secara bergiliran ke setiap server dalam grup. Setelah server menerima permintaan, kemudian dipindahkan ke akhir antrian dan server berikutnya dalam urutan menerima permintaan selanjutnya. Pendekatan ini memastikan bahwa semua server menerima jumlah permintaan yang sama dalam jangka waktu tertentu, asalkan semua permintaan memiliki kebutuhan sumber daya yang serupa [19]. Oleh karena itu, *Round-Robin* bisa menjadi pilihan yang baik dalam lingkungan di mana beban kerja merata [20].

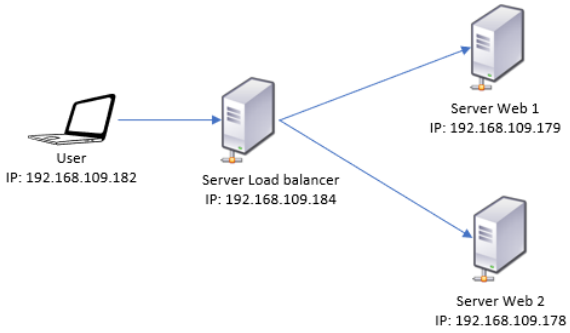
Algoritma ini mengoperasikan konsep pengaturan giliran, di mana setiap permintaan dari klien dialokasikan secara bergantian ke setiap server yang tersedia. Proses ini dilakukan dalam urutan siklus putaran, di mana setiap server menerima sejumlah permintaan sebelum beralih ke server berikutnya dalam daftar. Pendekatan ini memastikan bahwa beban kerja terdistribusi secara merata di antara semua server yang terlibat, sehingga mencegah server tertentu mengalami beban yang berlebihan sementara yang lain tidak terpakai. Namun demikian, *Round-Robin* juga memiliki beberapa keterbatasan. Misalnya, jika beban kerja tidak merata, yaitu beberapa permintaan memerlukan lebih banyak sumber daya daripada lainnya, metode ini mungkin tidak efisien karena dapat

P-ISSN : 2089-3353
E-ISSN : 2808-9162

menyebabkan beberapa server menjadi terbebani berlebihan.

3. Hasil dan Pembahasan

Pendekatan *load balancing* diterapkan dengan mendistribusikan permintaan atau tugas di antara berbagai server atau sumber daya komputasi, sehingga tidak ada satu entitas pun yang mengalami beban kerja yang berlebihan sesuai dengan arsitektur serta rancangan yang telah direncanakan sebelumnya dengan algoritma *Least-Connection* dan *Round-Robin* dalam lingkungan simulasi *cloud*. Selanjutnya dilakukan evaluasi dengan berbagai skenario beban kerja untuk mengevaluasi performa kedua teknik tersebut dalam kondisi yang berbeda. Dalam penelitian yang menilai efektivitas dari *Round-Robin*, diperoleh hasil r*esponse time* dalam milidetik yang terukur untuk lima tingkat koneksi yang berbeda. Gambar 3 merupakan rancangan topologi dari skema pengujian yang akan diterapkan.



Gambar 3. Arsitektur Server Load Balancer

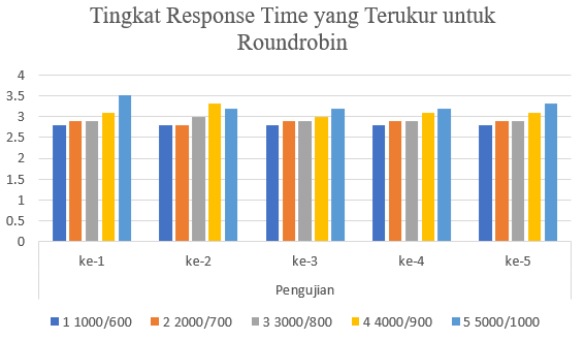
Pada tingkat koneksi pertama, 1000/600, waktu respons tercatat konstan pada 2.8 ms untuk semua lima kali pengujian, menghasilkan rata-rata yang sama dengan nilai pengujian individu. Menariknya, meskipun tingkat koneksi meningkat menjadi 2000/700, peningkatan *response time* relatif minim, dengan rata-rata hanya sedikit naik menjadi 2.875 ms. Pada tingkat koneksi 3000/800, kita melihat konsistensi yang sama dengan sedikit variasi, di mana pengujian kedua mencatat sedikit lonjakan waktu respons menjadi 3 ms, namun rata-rata tetap di bawah 3 ms, yaitu 2.925 ms. Menariknya, ketika tingkat koneksi meningkat menjadi 4000/900, ada peningkatan yang lebih signifikan dalam waktu respon dengan pengujian kedua yang mengalami puncak pada 3.3 ms, menghasilkan rata-rata 3.125 ms. Ini mungkin mengindikasikan bahwa semakin tinggi beban koneksi, sistem mengalami lebih banyak tekanan yang mencerminkan dalam peningkatan waktu respons. Pada tingkat tertinggi yang diuji, yaitu 5000/1000, kita melihat rata-rata waktu respons tertinggi yaitu 3.275 ms.

Walaupun ada fluktuasi di pengujian individu, dengan pengujian pertama menunjukkan waktu respons paling tinggi pada 3.5 ms, sistem masih menunjukkan kemampuan untuk menangani beban yang lebih tinggi dengan peningkatan waktu respons yang tidak terlalu drastis. Keseluruhan data menunjukkan bahwa metode *Round-Robin* mampu mempertahankan waktu respons yang relatif stabil bahkan ketika tingkat koneksi meningkat, namun ada indikasi peningkatan waktu respons seiring dengan penambahan beban. Ini menunjukkan bahwa metode *Round-Robin* menyediakan distribusi beban yang efisien namun memiliki batasan dalam mengelola kenaikan beban yang tinggi secara efektif. Tabel 3 berisikan hasil pengujian *response time* (ms) Server dengan *Round-Robin*.

Tabel 4. Hasil Pengujian *Response Time* Dengan *Round-Robin*

No	Tingkat Koneksi	Pengujian					Rata-Rata
		1	2	3	4	5	
1	1000/600	2.8	2.8	2.8	2.8	2.8	2.8
2	2000/700	2.9	2.8	2.9	2.9	2.9	2.875
3	3000/800	2.9	3	2.9	2.9	2.9	2.925
4	4000/900	3.1	3.3	3	3.1	3.1	3.125
5	5000/1000	3.5	3.2	3.2	3.2	3.3	3.275

Berdasarkan data yang diperoleh dari pengujian *load balancing* dengan metode *Least-Connection*, kita dapat melihat bagaimana sistem berperilaku terhadap berbagai tingkat koneksi. Pada tingkat koneksi awal 1000/600, terdapat sedikit variasi dalam waktu respons, dengan nilai minimum 2.8 ms dan maksimum 3 ms, menghasilkan rata-rata 2.9 ms. Ini menunjukkan bahwa pada beban rendah, sistem dengan metode *Least-Connection* mampu mempertahankan konsistensi. Untuk tingkat koneksi 2000/700, hasil pengujian menunjukkan kestabilan yang sempurna tanpa variasi, dengan waktu respons konstan pada 2.9 ms untuk semua lima kali pengujian, memberikan gambaran bahwa sistem dapat menangani peningkatan beban tanpa mempengaruhi performa waktu respons. Grafik batang pada Gambar 4 memperlihatkan tingkat *response time* yang didapat dari pengujian *Round Robin*.



Gambar 4. Arsitektur Server Load Balancer

Ketika tingkat koneksi meningkat menjadi 3000/800, kita mulai melihat sedikit peningkatan pada waktu respons, terutama pada pengujian keempat dan kelima yang menunjukkan waktu respons 3 ms dan 3.1 ms, secara berurutan. Hal ini memberikan rata-rata waktu respons sebesar 2.96 ms, yang masih tergolong efisien namun menandakan adanya peningkatan beban yang

P-ISSN : 2089-3353
E-ISSN : 2808-9162

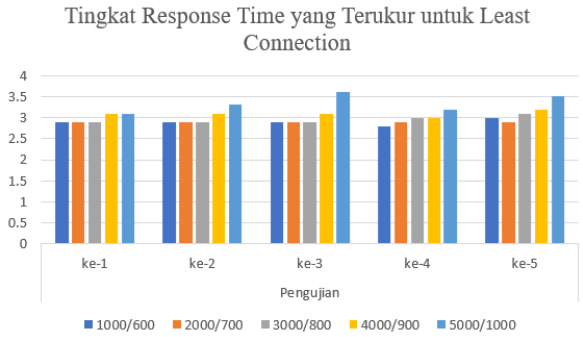
dimulai mempengaruhi performa. Lebih jauh lagi, pada tingkat koneksi 4000/900, rata-rata waktu respons naik menjadi 3.1 ms. Penambahan beban ini tampaknya memiliki pengaruh yang lebih jelas, dengan waktu respons yang berfluktuasi antara 3 ms hingga 3.2 ms. Meskipun terjadi peningkatan, sistem masih menunjukkan kemampuan yang baik dalam menangani peningkatan beban dengan metode *Least-Connection*.

Terakhir, pada tingkat koneksi tertinggi yang diuji, 5000/1000, rata-rata waktu respons tercatat sebesar 3.34 ms. Pengujian individu menunjukkan variasi yang lebih besar dari pengujian sebelumnya, dengan nilai tertinggi mencapai 3.6 ms. Ini dapat menunjukkan bahwa pada tingkat koneksi tinggi, metode *Least-Connection* mulai menemui keterbatasan dalam memproses permintaan secara efektif, yang tercermin dari peningkatan waktu respons. Tabel 4 berisikan hasil pengujian *response time* (ms) Server dengan *Least-Connection*.

Tabel 5. Hasil Pengujian *Response time* Dengan *Least-Connection*

No	Tingkat Koneksi	Pengujian					Rata-Rata
		1	2	3	4	5	
1	1000/600	2.9	2.9	2.9	2.8	3	2.9
2	2000/700	2.9	2.9	2.9	2.9	2.9	2.9
3	3000/800	2.9	2.9	2.9	3	3.1	2.96
4	4000/900	3.1	3.1	3.1	3	3.2	3.1
5	5000/1000	3.1	3.3	3.6	3.2	3.5	3.34

Dari tabel yang diberikan, kita dapat melakukan perbandingan hasil rata-rata pengujian waktu respons (*response time*) antara metode *load balancing Round-Robin* dan *Least-Connection* pada berbagai tingkat koneksi. Grafik batang pada Gambar 5 memperlihatkan tingkat *response time* yang didapat dari pengujian *least connection.*



Gambar 5. Arsitektur Server Load Balancer

Tabel 6. Hasil Rata-Rata Uji *Response time* Pada Kedua Model

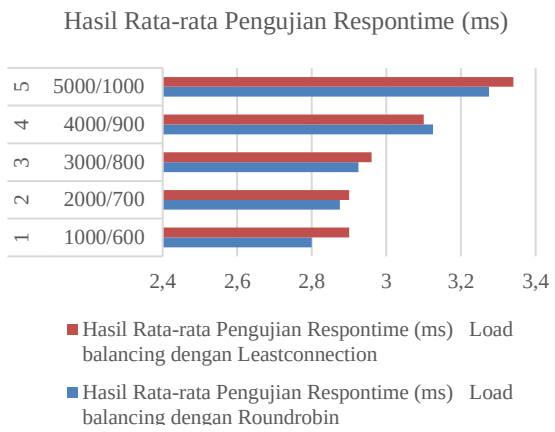
No	Tingkat Koneksi	Hasil Rata-Rata *Response time*	
		Round-Robin	*Least-Connection*
1	1000/600	2.8	2.9
2	2000/700	2.875	2.9
3	3000/800	2.925	2.96
4	4000/900	3.125	3.1
5	5000/1000	3.275	3.34

Tabel 5 diatas memperlihatkan hasil rata-rata pengujian *response time (ms)* pada kedua model. Pada tingkat koneksi 1000/600, metode *Round-Robin* mencatatkan waktu respons rata-rata yang sedikit lebih baik (2.8 ms) dibandingkan dengan metode *Least-Connection* (2.9 ms). Hal ini menunjukkan bahwa pada koneksi rendah, *Round-Robin* mungkin sedikit lebih efisien dalam mendistribusikan beban. Ketika tingkat koneksi meningkat menjadi 2000/700, metode *Round-Robin* masih menunjukkan waktu respons yang relatif konsisten (2.875 ms), sementara *Least-Connection* tetap konstan pada 2.9 ms. Meskipun perbedaannya tidak signifikan, hal ini dapat menandakan bahwa *Round-Robin* memiliki keunggulan ringan dalam menangani beban pada level ini.

Pada tingkat koneksi 3000/800, metode *Round-Robin* mengalami kenaikan waktu respons menjadi 2.925 ms, sementara metode *Least-Connection* mencatatkan waktu yang sedikit lebih tinggi (2.96 ms). Perbedaan waktu respons yang lebih nyata di tingkat ini dapat mengindikasikan bahwa beban yang lebih tinggi mulai memberikan dampak yang lebih besar terhadap performa sistem dengan metode *Least-Connection*. Untuk tingkat koneksi 4000/900, kita melihat pembalikan situasi di mana metode *Least-Connection* (3.1 ms) justru menunjukkan waktu respons rata-rata yang lebih baik dibandingkan dengan *Round-Robin* (3.125 ms). Hal ini bisa menandakan bahwa pada beban yang lebih berat, *Least-Connection* mulai lebih efektif dalam menyeimbangkan permintaan dibandingkan dengan *Round-Robin*. Pada koneksi tertinggi yang diuji, 5000/1000, *Round-Robin* mengalami waktu respons rata-rata sebesar 3.275 ms, sementara *Least-Connection* berada pada 3.34 ms. Pada tahap ini, perbedaan menjadi lebih jelas dimana *Round-Robin* tampaknya lebih tangguh dalam menghadapi kenaikan beban dibandingkan dengan *Least-Connection*.

Dari hasil pengujian yang telah dilakukan, kedua metode menunjukkan kekuatan dan kelemahan pada berbagai tingkat koneksi. *Round-Robin* cenderung lebih konsisten pada tingkat koneksi rendah hingga sedang, sedangkan *Least-Connection* menunjukkan potensi efisiensi yang lebih baik pada koneksi menengah hingga tinggi. Namun, pada tingkat beban yang sangat tinggi, *Round-Robin* tampaknya memiliki keunggulan dalam mempertahankan waktu respons yang lebih rendah. Dalam konteks aplikasi nyata, pemilihan metode *load balancing* dapat tergantung pada pola lalu lintas yang diharapkan dan prioritas sistem, apakah itu untuk konsistensi waktu respons atau untuk fleksibilitas dalam menangani lonjakan beban yang tidak terduga.

P-ISSN : 2089-3353
E-ISSN : 2808-9162



Gambar 6. Hasil Nilai Waktu Respons Antara Metode Server Tunggal dan *Load balancing*

Grafik batang pada Gambar 5 menunjukkan hasil Rata-rata pengujian *response time* untuk dua metode yaitu *load balancing* dengan *Least-Connection* dan *Round-Robin*. Data disajikan untuk lima tingkat koneksi yang berbeda dari 1000/600 hingga 5000/1000.

4. Kesimpulan

Dalam konteks *cloud*, metode *Round-Robin* efektif untuk mengatur trafik ringan dan memberikan waktu respons yang stabil, cocok untuk aplikasi dengan trafik yang dapat diprediksi. Sebaliknya, *Least-Connection* lebih unggul dalam mengelola trafik padat, yang penting untuk aplikasi yang mengalami lonjakan permintaan. Kesimpulan yang dapat ditarik adalah bahwa *load balancing* dengan *Least-Connection* mampu mengelola permintaan dengan efektif hingga tingkat tertentu sebelum performanya mulai terdegradasi dengan peningkatan beban. Waktu respons yang masih berada di bawah ambang 4 ms hingga tingkat koneksi 5000/1000 menunjukkan bahwa metode ini cukup efektif, namun ada ruang untuk optimasi lebih lanjut untuk menangani peningkatan beban dengan lebih efisien. Perbandingan langsung dengan data dari metode *Round-Robin* dapat memberikan wawasan tentang kelebihan dan keterbatasan kedua metode dalam berbagai skenario beban kerja. Pilihan strategi *load balancing* harus disesuaikan dengan mempertimbangkan pola trafik dan kebutuhan spesifik layanan *cloud* untuk mencapai keseimbangan antara efisiensi sumber daya dan keandalan layanan.

Daftar Rujukan

[1] 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.

[2] A. Oad, K. Kumari, I. Hussain, F. Dong, B. Hammad, and R. Oad, "Performance comparison of ORB, SURF and SIFT using Intracranial Haemorrhage CTScan Brain images," *Int. J. Artif. Intell. Math. Sci.*, vol. 1, no. 2, pp. 26–34, 2023, doi: 10.58921/ijaims.v1i2.41.

[3] Z. Bustomi, M. Syahiruddin, M. I. Afandi, and K. F. H. Holle, "Load Balancing Web Server Menggunakan Nginx pada Lingkungan Virtual," *J. Inform. J. Pengemb. IT*, vol. 5, no. 1, pp. 32–36, 2019.

[4] S. Goswami and A. De Sarkar, "A comparative study of load balancing algorithms in computational grid environment," *Proc. Int. Conf. Comput. Intell. Model. Simul.*, pp. 99–104, 2013, doi: 10.1109/CIMSim.2013.24.

[5] 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.

[6] T. S. Hendrana and I. M. Suartana, "Penerapan Container Load Balancing untuk Manajemen Trafik pada Learning Management System," *JINACS J. Informatics Comput. Sci.*, vol. 04, no. 02, pp. 169–182, 2022.

[7] Vignesh Joshi, "Load Balancing Algorithms in Cloud Computing," *Int. J. Res. Eng. Innov.*, vol. 3, no. 6, pp. 530–532, 2020, doi: 10.1007/s10586-019-02928-y.

[8] N. Rachmatullah, D. Mukarromah, and T. Sutabri, "Learning Management System Berbasis Cloud dalam Model Pembelajaran Blended Learning Pada Fakultas Saintek UIN Raden Fatah," *J. Fasilkom*, vol. 13, no. 02, pp. 132–137, 2023, doi: 10.37859/jf.v13i02.5024.

[9] Y. Afrianto and A. H. Hendrawan, "Implementasi Data Center Untuk Penempatan Host Server Berbasis Private Cloud Computing," *Krea-Tif*, vol. 7, no. 1, pp. 50–59, 2019, doi: 10.32832/kreatif.v7i1.2031.

[10] 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.

[11] P. Singh, P. Baaga, and S. Gupta, "Assorted Load Balancing Algorithms in Cloud Computing: A Survey," *Int. J. Comput. Appl.*, vol. 143, no. 7, pp. 34–40, 2016, doi: 10.5120/ijca2016910258.

[12] J. Vashistha, "Comparative Study of Load Balancing Algorithms," *IOSR J. Eng.*, vol. 03, no. 03, pp. 45–50, 2013, doi: 10.9790/3021-03324550.

[13] 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.

[14] Q. Zhang, L. Cheng, and R. Boutaba, "Cloud computing: State-of-the-art and research challenges," *J. Internet Serv. Appl.*, vol. 1, no. 1, pp. 7–18, 2010, doi: 10.1007/s13174-010-0007-6.

[15] W. Zhang, "Linux Virtual Server for Scalable Network Services," *Ottawa Linux Symp.*, 2000.

[16] 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.

[17] 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.

[18] 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.

[19] 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, doi: 10.30743/infotekjar.v4i1.1688.

[20] 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, no. 2, pp. 9669–9682, 2022.