JURNAL TEKNOINFO
Volume 18, Nomor 1, Januari 2024, Page 173-181
ISSN: 1693-0010(Print), ISSN: 2615-224X(Online)
Available online at https://ejurnal.teknokrat.ac.id/index.php/teknoinfo/index

ANALISIS PERFORMA LOAD BALANCING TERHDAP THROUGHPUT PADA KLUSTER SERVER UNTUK MENDUKUNG SMART CITY

Wartono[1)], Rudiansyah[2)], M. Hadi Prayitno[3)], Joko Trianto[4)]

[1] Sistem Informasi Kota Cerdas, Universitas Tunas Pembangunan
[2] Sistem Informasi, Universitas Sjakhyakirti
[3] Informatika, Universitas Bhayangkara Jakarta Raya
[4] Teknik Informatika, Sekolah Tinggi Teknologi Informasi NIIT
[1] Jl. Walanda Maramis No.31, Nusukan, Banjarsari, Kota Surakarta, Jawa Tengah, Indonesia
[2] Jl. Sultan Muhammad Mansyur Kb Gede, 32 Ilir, Ilir Barat II, Kota Palembang, Sumatera Selatan, Indonesia
[3] Jl. Harsono RM No.67, Ragunan, Pasar Minggu, Jakarta Selatan, DKI Jakarta, Indonesia
[4] 1Jl. Asem Dua No.22, Cipete Selatan, Cilandak, Kota Jakarta Selatan, DKI Jakarta, Indonesia
Email: [1]wartono@lecture.utp.ac.id, [2]rudiansyah@unisti.ac.id, [3]hadi.prayitno@dsn.ubharajaya.ac.id, [4]jokotrianto.trianto@gmail.com

Abstract

In the current digital era, the demand for fast and efficient web-based services is increasing, especially in the context of smart cities that are highly dependent on information technology. Load balancing is an important solution to distribute workload evenly between servers, thereby maximizing throughput and reducing response time. This research uses an experimental method by testing load balancing performance on a cloud computing infrastructure model. Testing was carried out by comparing models with and without load balancing to see the system's ability to increase throughput. These measurements are carried out both before and after successful implementation of the load balancing system. As the basis of server infrastructure, virtual machines on the main computer are used to establish a load balancing system. The results show that the application of the load balancing model is better than the model without load balancing. This can be seen from the test results if the connection level increases, especially above 9000/900, then the system with load balancing is able to maintain stable and higher throughput compared to the system without load balancing. The findings show that increasing throughput contributes significantly to the efficiency and speed of web services. Implementing appropriate load balancing strategies can increase throughput and have a positive impact on public web services in the context of smart cities. Apart from that, the implementation of load balancing is able to process all incoming requests efficiently and helps achieve good quality of service (QoS) values.

Keyword: *cloud computing, load balancing, response time, smart city, quality of services.*

Abstrak

Dalam era digital saat ini, permintaan layanan berbasis web yang cepat dan efisien semakin meningkat, terutama dalam konteks kota cerdas yang sangat bergantung pada teknologi informasi. *Load balancing* menjadi solusi penting untuk mendistribusikan beban kerja secara merata di antara server-server, sehingga dapat memaksimalkan *throughput* dan mengurangi waktu respons. Penelitian ini menggunakan metode eksperimental dengan melakukan pengujian performa *load balancing* pada model infrastruktur *cloud computing*. Pengujian dilakukan dengan membandingkan model dengan dan tanpa *load balancing* untuk melihat kemampuan sistem dalam meningkatkan *throughput*. Pengukuran ini dilakukan baik sebelum maupun setelah implementasi sistem penyeimbangan beban berhasil. Sebagai dasar infrastruktur server, mesin virtual pada komputer utama digunakan untuk mendirikan sistem penyeimbangan beban. Hasilnya menunjukkan bahwa penerapan model *load balancing* lebih baik dibandingkan model tanpa *load balancing*. Ini terlihat dari pada hasil pengujian jika tingkat koneksi meningkat terutama di atas 9000/900, maka sistem dengan *load balancing* mampu mempertahankan *throughput* yang stabil dan lebih tinggi dibandingkan dengan sistem tanpa *load balancing*. Temuan menunjukkan bahwa peningkatan *throughput* berkontribusi signifikan terhadap efisiensi dan kecepatan layanan web. Implementasi strategi *load balancing* yang tepat dapat meningkatkan *throughput* dan berdampak positif pada pelayanan web publik dalam konteks kota cerdas. Selain itu implementasi *load balancing* mampu memproses semua permintaan masuk secara efisien dan membantu mencapai nilai *quality of service* (QoS) yang baik.

Kata Kunci: *cloud computing, load balancing*, waktu respon, kota cerdas, *quality of services*

JURNAL TEKNOINFO
Volume 18, Nomor 1, Januari 2024, Page 173-181
ISSN: 1693-0010(Print), ISSN: 2615-224X(Online)
Available online at https://ejurnal.teknokrat.ac.id/index.php/teknoinfo/index

1. PENDAHULUAN

Perkembangan teknologi informasi dan komunikasi (TIK) telah membawa perubahan signifikan dalam berbagai aspek kehidupan manusia. Salah satu aplikasi penting dari TIK adalah dalam pengembangan konsep kota cerdas atau "*smart city*". Kota cerdas adalah suatu konsep perkotaan yang menggunakan teknologi digital dan data untuk meningkatkan kualitas hidup warganya, mengoptimalkan operasional kota, serta mendorong pertumbuhan ekonomi dan inovasi [1]. Dalam kota cerdas, sistem-sistem yang berbeda, seperti transportasi, energi, infrastruktur, dan pelayanan publik, terintegrasi secara pintar dan efisien [2]. Tentunya, pada implementasi sistem informasi di kota cerdas, performa layanan berbasis web menjadi faktor yang sangat penting. Hal ini dikarenakan banyaknya layanan publik yang sudah beralih ke *platform digital* [3]. Dengan semakin meningkatnya permintaan terhadap layanan-layanan ini, efisiensi dan kecepatan layanan menjadi hal yang harus diperhatikan [4]. Salah satu cara untuk meningkatkan efisiensi dan kecepatan layanan berbasis web adalah melalui optimisasi infrastruktur teknologi informasi dengan memaksimalkan *throughput* serta meminimalisir waktu respons. *Throughput* dalam jaringan komputer merujuk pada seberapa banyak data yang dapat dikirimkan atau diterima melalui jaringan dalam satuan waktu tertentu [5]. Ini adalah salah satu metrik penting yang digunakan untuk mengukur kinerja jaringan. Salah satu metode yang bisa digunakan untuk mencapai tujuan tersebut adalah *load balancing*. *Load balancing* adalah proses mendistribusikan beban kerja secara merata di antara server-server dalam suatu sistem sehingga dapat memaksimalkan *throughput* dan mengurangi waktu respons [6], [7]. Tujuan utamanya adalah untuk meningkatkan kinerja, ketersediaan, dan keandalan sistem, serta mencegah kelebihan beban pada satu sumber daya yang dapat menyebabkan gangguan atau penurunan kinerja [8], [9]. *Load balancing* juga dapat membantu dalam menjaga stabilitas sistem dengan mencegah *overloading* pada salah satu server [10].

Teknik *load balancing* digunakan dalam sejumlah penelitian untuk mendistribusikan beban kerja server dan memberikan kinerja yang lebih optimal. Studi pertama meneliti penggunaan strategi penyeimbangan beban pada server untuk meningkatkan fungsionalitas situs web untuk pembelajaran online [11]. Pada penelitian ini metode loadbalancing diuji pada 500 koneksi dengan 10 permintaan per detik, menghasilkan nilai waktu respon sebesar 36,4 ms. Penyelidikan tambahan terhadap penerapan teknik penyeimbangan beban pada server web mengungkapkan bahwa teknik ini mampu menangani permintaan yang salah [12]. Sepuluh ribu pertanyaan digunakan dalam pengujian waktu respons langsung penelitian ini. Penggunaan loadbalancing untuk meningkatkan jaringan internet pada jaringan dengan lalu lintas padat adalah subjek penelitian yang akan datang [13]. Studi ini menunjukkan bahwa teknik yang digunakan dapat mengakibatkan penundaan antara 21 dan 24 ms.

Namun demikian, pemilihan algoritma *load balancing* harus dilakukan dengan hati-hati karena berbagai algoritma memiliki karakteristik dan efektivitas yang berbeda-beda. Performa dari algoritma *load balancing* bisa sangat bervariasi tergantung pada konteks penggunaannya [14]. Oleh karena itu, penelitian ini bertujuan untuk menganalisis performa dari beberapa algoritma *load balancing* populer dalam konteks kota cerdas serta mengevaluasi pengaruhnya terhadap *throughput*. Selain itu, penelitian ini juga bertujuan untuk melihat dampak dari peningkatan *throughput* terhadap pelayanan publik di lingkungan kota cerdas. Implementasi strategi *load balancing* yang tepat dapat memberikan manfaat besar bagi sistem informasi di lingkungan kota cerdas. Pada arsitektur kota cerdas pengelolaan distribusi beban atau tugas-tugas yang ada di seluruh infrastruktur teknologi informasi dan komunikasi (TIK) sangat membantu dalam memberi dukungan terwujudnya kota cerdas [15]. Karena melalui *load balancing* dapat mendistribusikan beban kerja (*traffic*) secara merata di antara berbagai perangkat atau server dalam jaringan [16], [17]. Walaupun demikian, masih ada beberapa tantangan yang perlu diatasi, seperti masalah keamanan data dan privasi pengguna [18]. Dengan adanya *load balancing* dapat mencegah terjadinya ketidakseimbangan beban kerja yang dapat mengakibatkan server menjadi overused, mengalami kelebihan beban, atau bahkan kegagalan.

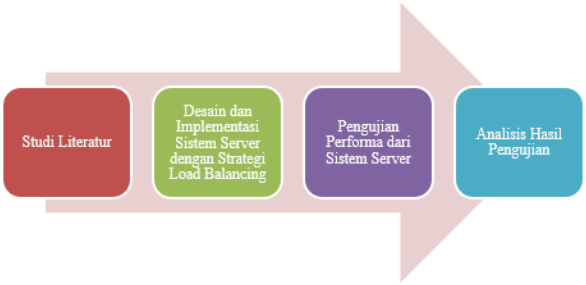
Dengan melihat pentingnya isu ini, penelitian ini berusaha untuk memberikan kontribusi dalam memahami lebih lanjut tentang bagaimana performa *load balancing* dapat ditingkatkan dan bagaimana dampaknya terhadap *throughput* dalam konteks kota cerdas. Hasil dari penelitian ini diharapkan dapat digunakan sebagai referensi bagi pengambil kebijakan dan praktisi dalam merancang dan mengimplementasikan strategi *load balancing* di lingkungan kota cerdas.

2. METODE PENELITIAN

2.1 Tahapan Penelitian

Penelitian ini dilakukan dalam beberapa tahapan, sebagaimana diperlihatkan pada Gambar 1 dengan melibatkan pengujian terhadap metode *load balancing* untuk mendistribusikan sejumlah besar koneksi yang masuk ke setiap server web.

JURNAL TEKNOINFO
Volume 18, Nomor 1, Januari 2024, Page 173-181
ISSN: 1693-0010(Print), ISSN: 2615-224X(Online)
Available online at https://ejurnal.teknokrat.ac.id/index.php/teknoinfo/index



Gambar 1. Alur penelitian.

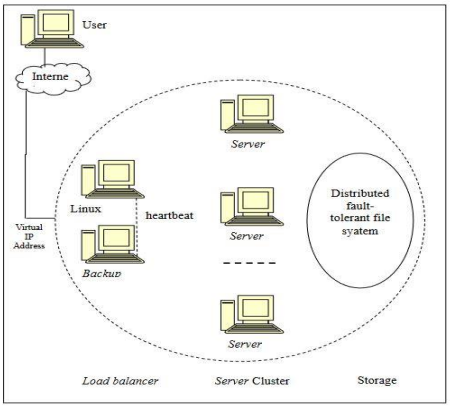
1) Tahap pertama adalah studi literatur, dimana peneliti melakukan penelaahan mendalam terhadap berbagai sumber literatur yang relevan dengan topik penelitian, seperti jurnal-jurnal ilmiah dan buku-buku teks. Studi literatur ini membantu peneliti untuk memahami konteks dan latar belakang masalah yang diteliti.
2) Tahap kedua adalah desain dan implementasi sistem server dengan strategi *load balancing*. Sistem server ini dirancang dengan mempertimbangkan kebutuhan spesifik dari kota cerdas, seperti kapasitas data yang besar dan jumlah pengguna yang banyak.
3) Tahap ketiga adalah pengujian performa dari sistem server. Pengujian ini dilakukan dengan menggunakan skenario tertentu untuk mengevaluasi efektivitas dari strategi *load balancing* dalam meningkatkan *throughput*.
4) Tahap terakhir adalah analisis hasil pengujian. Hasil-hasil pengujian kemudian dianalisis untuk mengetahui sejauh mana performa sistem server dapat ditingkatkan melalui strategi *load balancing*.

2.2 Skenario Pengujian

Pengujian performa sistem server ini menggunakan skenario simulasi beban kerja berdasarkan pola akses layanan publik di lingkungan kota cerdas. Beban kerja tersebut kemudian didistribusikan ke berbagai server dalam sistem dengan memanfaatkan algoritma *load balancing*. Setiap skenario dijalankan beberapa kali untuk memastikan hasil yang andal. Performa *load balancing* selanjutnya dibandingkan dengan sistem tanpa *load balancing*, menggunakan metrik *throughput* dan pengujian pada berbagai tingkat koneksi. Pengujian pertama dilakukan dengan 1000 koneksi dan 100 permintaan per detik, yang kemudian ditingkatkan secara bertahap hingga mencapai 11000 koneksi dan 1100 permintaan per detik. Tujuan utama penggunaan *load balancing* dalam skenario ini adalah untuk mengevaluasi nilai *throughput* yang diperoleh. Load balancer mendistribusikan permintaan dari pengguna ke setiap server secara merata, sehingga beban kerja dapat tersebar merata di antara semua server [19], [20]. Hal ini diharapkan dapat menghasilkan nilai *throughput* yang optimal.

2.2 Skenario Pengujian

Sistem server dirancang sebagai sebuah model yang menyerupai infrastruktur *cloud computing* karena fleksibilitas dan skalabilitasnya. Infrastruktur ini terdiri dari beberapa node atau mesin virtual yang saling terhubung melalui jaringan komputer. Setiap node memiliki perangkat lunak tertentu yang bertugas mendistribusikan beban kerja kepada node-node lainnya sesuai dengan algoritma *load balancing* yang digunakan. Algoritma *load balancing* yang dipertimbangkan dalam penelitian ini adalah *Round Robin*.



Gambar 2. Arsitektur *Server Load Balancer*

JURNAL TEKNOINFO
Volume 18, Nomor 1, Januari 2024, Page 173-181
ISSN: 1693-0010(Print), ISSN: 2615-224X(Online)
Available online at https://ejurnal.teknokrat.ac.id/index.php/teknoinfo/index

Pada Gambar 2 menunjukkan arsitektur server *load balancer* yang merupakan infrastruktur *cloud computing* terdiri dari beberapa node atau mesin virtual yang saling terhubung melalui jaringan komputer. Komponen pertama dalam arsitektur ini adalah server penyeimbang beban, yang bertugas mendistribusikan beban koneksi ke masing-masing server web. Server penyeimbang beban memiliki alamat IP spesifik yang memfasilitasi koneksinya dengan setiap server aplikasi dan komputer klien. Komponen kedua adalah rangkaian server web. Dalam konteks penelitian ini, kami menggunakan dua unit sebagai server web. Masing-masing dari dua unit ini menjalankan program aplikasi web yang dapat diakses oleh pengguna. Komponen ketiga adalah penyimpanan data atau basis data. Dalam penelitian ini, basis data ditempatkan langsung pada masing-masing server web, bukan secara terpisah atau individu. Pengujian sistem ini dilakukan dalam lingkungan mesin virtual dengan spesifikasi sebagaimana ditampilkan pada Tabel 1. Tabel 1 menampilkan spesifikasi mesin virtual dan komputer utama yang digunakan dalam pengujian sistem ini. Spesifikasi tersebut mencakup detail tentang CPU, RAM, SSD, dan sistem operasi yang diinstal pada setiap komponen.

Tabel 1. Spesifikasi mesin virtual

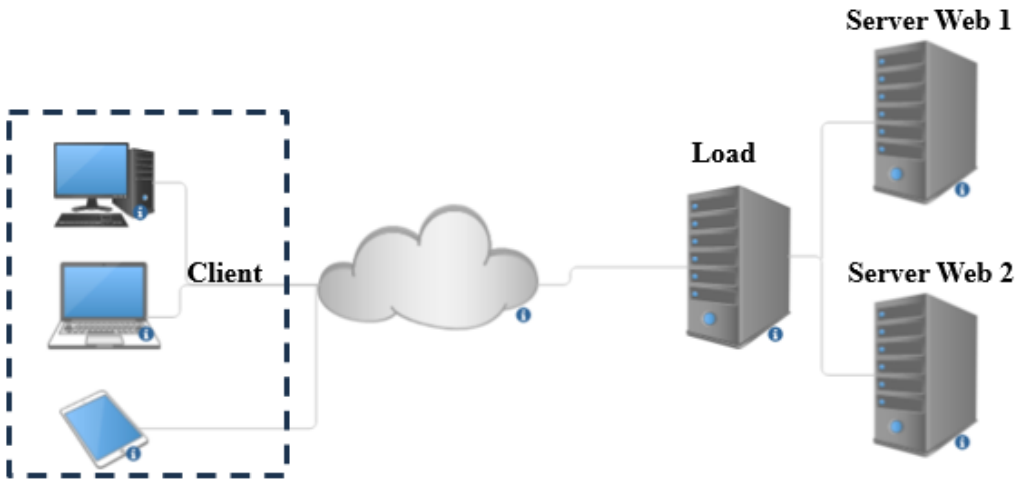
Komponen	CPU	RAM	SSD	Sistem Operasi
Komputer Utama	AMD Ryzen 5 (8 CPU) - 2,0 GHz	8 GB	250 GB	Windows 10
Mesin Virtual Load Balancer	AMD Ryzen 5 (1 CPU) - 2,0 GHz	1 GB	20 GB	Ubuntu Server
Mesin Virtual Server 1	AMD Ryzen 5 (1 CPU) - 2,0 GHz	1 GB	20 GB	Ubuntu Server
Mesin Virtual Server 2	AMD Ryzen (1 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

3. HASIL DAN PEMBAHASAN

Untuk mengevaluasi kinerja sistem server dalam mengelola sejumlah besar permintaan dari klien dalam jangka waktu tertentu, kami melakukan pengujian pada berbagai tingkat koneksi server *load balancing*. Parameter utama yang diukur dalam proses pengujian ini adalah *throughput*. Pengukuran ini dilakukan baik sebelum maupun setelah implementasi sistem penyeimbangan beban berhasil. Sebagai dasar infrastruktur server, mesin virtual pada komputer utama digunakan untuk mendirikan sistem penyeimbangan beban. Dalam studi ini, kami menganalisis dua model arsitektur server. Model pertama adalah arsitektur server tunggal, dimana sistem tidak menggunakan teknik penyeimbangan beban. Dalam model ini, hanya ada satu server yang bertugas menangani semua permintaan dari pengguna. Akibatnya, ketika jumlah permintaan meningkat secara signifikan, semua permintaan tersebut dialihkan ke satu server tersebut saja. Model kedua melibatkan implementasi serangkaian server yang didedikasikan untuk memproses permintaan masuk secara bersama-sama. Model ini menggunakan konsep mekanisme penyeimbangan beban dan terdiri dari sebuah server penyeimbang beban serta dua web servers. Berbeda dengan model pertama, setiap permintaan masuk dari klien diproses oleh dua atau lebih servers secara simultan. Oleh karena itu, studi ini mencakup evaluasi performa antara model arsitektur tunggal tanpa *load balancing* dan model dengan implementasi *load balancing* melalui serangkaian web servers.

Gambar 3. Model arsitektur dengan *server* tunggal

JURNAL TEKNOINFO
Volume 18, Nomor 1, Januari 2024, Page 173-181
ISSN: 1693-0010(Print), ISSN: 2615-224X(Online)
Available online at https://ejurnal.teknokrat.ac.id/index.php/teknoinfo/index



Gambar 4. Model arsitektur *load balancing* dengan dua buah server web.

Desain server tunggal dan sistem server yang melibatkan server penyeimbang beban, Server web 1, Server web 2, dan klien ditampilkan dalam Gambar 3 dan 4. Dalam konfigurasi ini, semua komponen berinteraksi satu sama lain dalam satu jaringan melalui switch. Klien dapat mengakses layanan melalui jaringan ini dan menerima layanan dari Server web 1 dan Server web 2.

Tabel 2. Perangkat dan Alamat IP

Perangkat	Alamat IP
Server Load balancing	192.168.109.184
Web Server 1	192.168.109.179
Web Server 2	192.168.109.178
Client	192.168.109.182

Pada Tabel 2 memperlihatkan alamat IP yang telah diatur untuk setiap perangkat dalam jaringan. Dari tabel tersebut, kita bisa melihat bahwa server penyeimbang beban memiliki alamat IP 192.168.109.184 dan bertugas mendistribusikan permintaan layanan dari klien ke masing-masing server web. Ketika klien mencoba mengakses layanan melalui server web, mereka secara otomatis dialihkan ke server penyeimbang beban, yang kemudian meneruskan permintaan tersebut ke salah satu dari server web di belakangnya. Klien tidak perlu mengetahui mana dari server web yang akan memproses permintaannya karena hal ini ditangani oleh server penyeimbang beban. Penelitian ini berfokus pada pengujian nilai *throughput* sebagai indikator kinerja layanan saat teknik *load balancing* diterapkan pada sistem server. Selama proses pengujian, klien menggunakan software benchmarking httperf untuk membuat sejumlah simultan koneksi dan mendapatkan nilai *throughput*. Hasil pengujian dapat dilihat pada Tabel 3 dan Tabel 4, yang menunjukkan hasil implementasi *load balancing* dengan rangkaian koneksi dari client ke web servers melalui software benchmarking httperf. Permintaan dilakukan secara bertahap, mulai dari 1000 kali dengan 100 permintaan/detik dan meningkat secara bertahap hingga mencapai 11000 kali dengan 1100 permintaan/detik. Nilai *throughput* dapat berbeda-beda pada setiap skenario pengujian karena durasi serta antrian proses unik tiap skenario; semakin banyak jumlah permintaan kepada web servers maka nilai *throughput* juga akan semakin meningkat. Berdasarkan nilai *throughput* ini, kita dapat mengevaluasi performa aplikasi dalam konteks kedua desain arsitektur sistem.

Tabel 3. Hasil pengujian *throughput (Kb/s)* Server dengan *load balancing*

No	Waktu Respon (ms) Server dengan Load balancing						
	Tingkat Koneksi	Pengujian ke- 1	Pengujian ke-2	Pengujian ke-3	Pengujian ke- 4	Pengujian ke-5	Hasil Rata-rata Pengujian
1	1000/100	1158.2	1158	1158.4	1157.9	1157.2	1157.94
2	2000/200	2315	2315.2	2315.5	2315.7	2315.2	2315.32

JURNAL TEKNOINFO
Volume 18, Nomor 1, Januari 2024, Page 173-181
ISSN: 1693-0010(Print), ISSN: 2615-224X(Online)
Available online at https://ejurnal.teknokrat.ac.id/index.php/teknoinfo/index

3	3000/300	3470.6	3471.9	3472	3472.4	3472.7	3471.92
4	4000/400	4629.5	4630	4628.3	4629.4	4629.8	4629.4
5	5000/500	5786.7	5785.3	5786.1	5784.5	5786.5	5785.82
6	6000/600	6943.8	6943.7	6941.9	6944	6942.4	6943.16
7	7000/700	8100.4	8101.1	8101.1	8101.7	8101.2	8101.1
8	8000/800	9256.9	9245.2	9258.6	9256	9258.1	9254.96
9	9000/900	10416.7	10409.5	10404.7	10416.1	10413.2	10412.04
10	10000/1000	11553.4	11570	11567.1	11573	11512.2	11555.14
11	11000/1100	12620	12603.4	12697.8	12625.3	12708.7	12651.04

Pada Tabel 3 menggambarkan hasil pengujian *throughput* (dalam Kb/s) dari server dengan *load balancing*. Pengujian dilakukan sebanyak lima kali pada setiap tingkat koneksi yang berbeda, mulai dari 1000/100 hingga 11000/1100. Pada tingkat koneksi 1000/100, nilai *throughput* berkisar antara 1157.2 dan 1158.4 Kb/s, dengan rata-rata hasil pengujian adalah 1157.94 Kb/s. Untuk tingkat koneksi selanjutnya yaitu 2000/200, nilai *throughput* berkisar antara 2315 dan 2315.7 Kb/s, dengan rata-rata hasil pengujian adalah 2315.32 Kb/s. Hal serupa terjadi pada tingkat koneksi yang lebih tinggi lagi; untuk contoh lainnya pada tingkat koneksi terakhir yaitu 11000/1100, nilai *throughput* berkisar antara12603.4 dan12708.7Kb/s,dengan rata-rata hasil pengujian adalah 12651.04Kb/s. Secara umum tabel ini menunjukkan bahwa semakin besar tingkat koneksi maka nilai *throughput* juga semakin besar, yang menunjukkan performa positif dari sistem server dengan *load balancing* dalam penanganan beban kerja yang meningkat..

Tabel 4. Hasil pengujian *throuhput (Kb/s)* Server tanpa *load balancing*

No	Waktu *Throughput* (Kb/s) Server tanpa Load balancing						
	Tingkat Koneksi	Pengujian ke-1	Pengujian ke-2	Pengujian ke-3	Pengujian ke- 4	Pengujian ke-5	Hasil Rata-rata Pengujian
1	1000/100	1157.8	1158.4	1157.9	1158.3	1158.2	1158.12
2	2000/200	2315.4	2315.2	2315.4	2315.3	2315.2	2315.3
3	3000/300	3473.1	3472.1	3472.2	3473	3472.1	3472.5
4	4000/400	4630	4629.2	4630.2	4629.2	4629.4	4629.6
5	5000/500	5785.8	5787.5	5785.2	5787	5786.9	5786.48
6	6000/600	6944	6945.3	6943.1	6943.4	6942.9	6943.74
7	7000/700	8101.4	8101.6	8102.5	8102.4	8100.4	8101.66
8	8000/800	9259.3	9257.1	9256.2	9257.9	9259	9257.9
9	9000/900	10413.7	10392.8	10371.7	10417.6	10415.8	10402.32
10	10000/1000	11305.5	8712.5	11574.8	8580	11573.4	10349.24
11	11000/1100	6739.8	7141.3	9658.9	7767.5	6874	7636.3

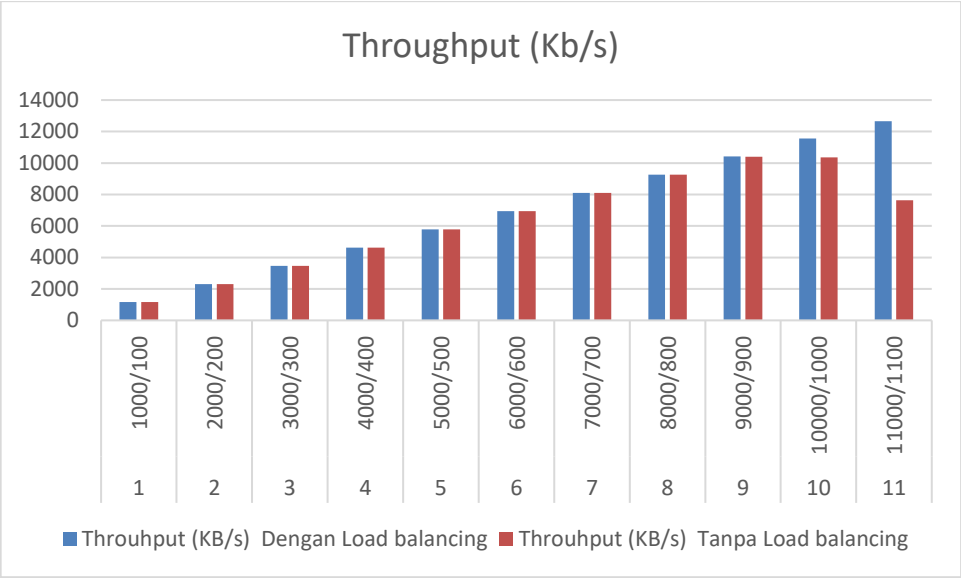
Pada Tabel 4 menampilkan hasil pengujian *throughput* (dalam Kb/s) dari server tanpa *load balancing*. Seperti tabel sebelumnya, pengujian dilakukan sebanyak lima kali pada setiap tingkat koneksi yang berbeda, mulai dari 1000/100 hingga 11000/1100. Pada tingkat koneksi 1000/100, nilai *throughput* berkisar antara 1157.8 dan 1158.4 Kb/s, dengan rata-rata hasil pengujian adalah 1158.12 Kb/s. Untuk tingkat koneksi selanjutnya yaitu 2000/200, nilai *throughput* berkisar antara 2315.2 dan 2315.4 Kb/s, dengan rata-rata hasil pengujian adalah 2315.3 Kb/s. Namun, terlihat perbedaan signifikan pada tingkat koneksi lebih tinggi seperti pada level 9000/900 hingga level tertinggi di tabel ini yaitu level koneksi11000/1100 dimana terjadi penurunan *throughput* yang cukup drastis dibandingkan dengan server menggunakan *load balancing*. Misalnya saja pada tingkat koneksi terakhir yaitu11000/1100, nilai *throughput* berkisar antara 6739.8 dan 9658.9Kb/s, dengan rata-rata hasil pengujian adalah 7636.3 Kb/s, yang jauh lebih rendah dibandingkan dengan server menggunakan *load balancing* pada level koneksi yang sama (12651.04Kb/s). Secara keseluruhan tabel ini menunjukkan

JURNAL TEKNOINFO
Volume 18, Nomor 1, Januari 2024, Page 173-181
ISSN: 1693-0010(Print), ISSN: 2615-224X(Online)
Available online at https://ejurnal.teknokrat.ac.id/index.php/teknoinfo/index

bahwa performa sistem server tanpa *load balancing* cenderung tidak stabil dan menurun saat menghadapi beban kerja yang meningkat.

Tabel 5. Hasil rata-rata pengujian *Throughput (Kb/s)* Server pada kedua arsitektur

No	Tingkat Koneksi	Rata-rata *Throughput* (Kb/s)	
		Dengan Load balancing	Tanpa Load balancing
1	1000/100	1157.94	1158.12
2	2000/200	2315.32	2315.3
3	3000/300	3471.92	3472.5
4	4000/400	4629.4	4629.6
5	5000/500	5785.82	5786.48
6	6000/600	6943.16	6943.74
7	7000/700	8101.1	8101.66
8	8000/800	9254.96	9257.9
9	9000/900	10412.04	10402.32
10	10000/1000	11555.14	10349.24
11	11000/1100	12651.04	7636.3



Gambar 5. Hasil nilai *throughput* antara metode server tunggal dan *load balancing*

Pada Gambar 5 menampilkan grafik perbandingan antara metode load balancing dengan single server. Hasil pengujian ini dapat dilihat berdasarkan nilai *throughput* yang diperoleh, seperti yang tercantum dalam Tabel 5. Tabel 5 ini menyajikan perbandingan rata-rata *throughput* (dalam Kb/s) antara server dengan *load balancing* dan tanpa *load balancing* pada berbagai tingkat koneksi, mulai dari 1000/100 hingga 11000/1100. Pada tingkat koneksi rendah hingga sedang (1000/100 hingga 8000/800), rata-rata *throughput* untuk kedua jenis server tampaknya serupa. Misalnya, pada tingkat koneksi 2000/200, rata-rata *throughput* untuk server dengan dan tanpa *load balancing* adalah sebesar 2315.32 Kb/s dan 2315.3 Kb/s secara berturut-turut. Namun, perbedaan yang signifikan terlihat pada tingkat koneksi yang lebih tinggi (9000/900 ke atas). Pada tingkat koneksi ini, server dengan *load balancing* menunjukkan nilai rata-rata *throughput* yang jauh lebih besar dibandingkan dengan server tanpa *load balancing*. Misalnya saja pada level koneksi tertinggi di tabel ini yaitu level koneksi11000/1100, rata-rata *throughput* untuk server dengan *load balancing* adalah12651.04Kb/s,

JURNAL TEKNOINFO
Volume 18, Nomor 1, Januari 2024, Page 173-181
ISSN: 1693-0010(Print), ISSN: 2615-224X(Online)
Available online at https://ejurnal.teknokrat.ac.id/index.php/teknoinfo/index

sedangkan untuk server tanpa *load balancing* hanya sebesar 7636.3Kb/s. Secara keseluruhan, tabel ini menggambarkan bahwa performa sistem server dengan *load balancing* cenderung lebih stabil dan mampu mempertahankan nilai *throughput* yang lebih baik saat menghadapi beban kerja yang meningkat dibandingkan sistem tanpa *load balancing*.

```
loadbalancing@httperf:~$ httperf --server=192.168.109.184 --uri=/index.html --num-conns=11000 --rate
=1100
httperf --client=0/1 --server=192.168.109.184 --port=80 --uri=/index.html --rate=1100 --send-buffer=
4096 --recv-buffer=16384 --num-conns=11000 --num-calls=1
httperf: warning: open file limit > FD_SETSIZE; limiting max. # of open files to FD_SETSIZE
Maximum connect burst length: 33

Total: connections 11000 requests 11000 replies 11000 test-duration 10.091 s

Connection rate: 1090.1 conn/s (0.9 ms/conn, <=189 concurrent connections)
Connection time [ms]: min 2.1 avg 73.7 max 328.3 median 73.5 stddev 45.6
Connection time [ms]: connect 1.4
Connection length [replies/conn]: 1.000

Request rate: 1090.1 req/s (0.9 ms/req)
Request size [B]: 78.0

Reply rate [replies/s]: min 1081.9 avg 1083.0 max 1084.0 stddev 1.4 (2 samples)
Reply time [ms]: response 70.7 transfer 1.7
Reply size [B]: header 254.0 content 11523.0 footer 0.0 (total 11777.0)
Reply status: 1xx=0 2xx=11000 3xx=0 4xx=0 5xx=0

CPU time [s]: user 0.16 system 9.64 (user 1.6% system 95.5% total 97.1%)
Net I/O: 12620.0 KB/s (103.4*10^6 bps)

Errors: total 0 client-timo 0 socket-timo 0 connrefused 0 connreset 0
Errors: fd-unavail 0 addrunavail 0 ftab-full 0 other 0
```

Gambar 6. pengujian koneksi 11000/1100 menggunakan teknik *load balancing*

Seperti tampak pada Gambar 6, implementasi *load balancing* mampu memproses semua permintaan masuk secara efisien dan membantu mencapai nilai *quality of service* (QoS) yang baik. Namun, selain itu dilakukan pengujian dengan beban sebanyak 11.000 request pada tingkat koneksi sebesar1100 koneks/detik, sehingga dihasilkan fenomena seperti pada Gambar 7.

```
loadbalancing@httperf:~$ httperf --server=192.168.109.178 --uri=/index.html --num-conns=11000 --rate
=1100
httperf --client=0/1 --server=192.168.109.178 --port=80 --uri=/index.html --rate=1100 --send-buffer=
4096 --recv-buffer=16384 --num-conns=11000 --num-calls=1
httperf: warning: open file limit > FD_SETSIZE; limiting max. # of open files to FD_SETSIZE
Maximum connect burst length: 15

Total: connections 10316 requests 10316 replies 10316 test-duration 17.720 s

Connection rate: 582.2 conn/s (1.7 ms/conn, <=1022 concurrent connections)
Connection time [ms]: min 6.0 avg 786.8 max 15040.4 median 150.5 stddev 1647.8
Connection time [ms]: connect 569.2
Connection length [replies/conn]: 1.000

Request rate: 582.2 req/s (1.7 ms/req)
Request size [B]: 78.0

Reply rate [replies/s]: min 155.0 avg 686.1 max 982.3 stddev 460.9 (3 samples)
Reply time [ms]: response 215.8 transfer 1.8
Reply size [B]: header 254.0 content 11523.0 footer 0.0 (total 11777.0)
Reply status: 1xx=0 2xx=10316 3xx=0 4xx=0 5xx=0

CPU time [s]: user 0.34 system 16.98 (user 1.9% system 95.8% total 97.7%)
Net I/O: 6739.8 KB/s (55.2*10^6 bps)

Errors: total 684 client-timo 0 socket-timo 0 connrefused 0 connreset 0
Errors: fd-unavail 684 addrunavail 0 ftab-full 0 other 0
```

Gambar 7. Pengujian koneksi 11000/1100 menggunakan server tunggal

Pada Gambar 7 memperlihatkan fenomena lain saat melakukan pengujian dengan beban sebanyak 11.000 request pada tingkat koneksi sebesar1100 koneks/detik. Server tidak mampu memproses semua *request* yang diterima. Hal tersebut ditandai dengan terdeteksinya kesalahan yang tercatat selama proses pengujian.

JURNAL TEKNOINFO
Volume 18, Nomor 1, Januari 2024, Page 173-181
ISSN: 1693-0010(Print), ISSN: 2615-224X(Online)
Available online at https://ejurnal.teknokrat.ac.id/index.php/teknoinfo/index

4. KESIMPULAN

Berdasarkan hasil penelitian yang telah dilakukan, dapat disimpulkan bahwa penggunaan teknologi *load balancing* pada sistem server memiliki dampak positif terhadap performa server, khususnya dalam konteks *throughput*. Pada tingkat koneksi rendah hingga sedang, performa sistem server dengan dan tanpa *load balancing* tampak serupa. Namun, ketika tingkat koneksi meningkat terutama di atas 9000/900, maka sistem dengan *load balancing* mampu mempertahankan *throughput* yang stabil dan lebih tinggi dibandingkan dengan sistem tanpa *load balancing*. Ini menunjukkan bahwa teknologi *load balancing* efektif dalam menjaga stabilitas dan performa sebuah sistem server saat menghadapi jumlah permintaan yang besar. Meskipun demikian, penting untuk diingat bahwa implementasi teknologi ini mungkin memerlukan biaya tambahan dan menambah kompleksitas teknis. Penelitian ini hanya berfokus pada aspek *throughput* dari performa server. Untuk evaluasi yang lebih komprehensif, aspek-aspek lain seperti latensi dan reliabilitas juga perlu dipertimbangkan dalam penelitian mendatang

DAFTAR PUSTAKA

[1] H. A. Viola and A. R. Fitrianto, "The Strategy Smart City Development Concepts in Indonesia," *J. Public Policy*, vol. 8, no. 1, pp. 1–10, 2022.
[2] M. D. Lytras, A. Visvizi, and A. Sarirete, "Clustering Smart City Services: Perceptions, Expectations, Responses," *Sustainability*, vol. 11, no. 1669, pp. 1–19, 2019, doi: 10.3390/su11061669.
[3] B. W. Wirtz, W. M. Müller, and F. W. Schmidt, "Digital Public Services in Smart Cities – an Empirical Analysis of Lead User Preferences," *Public Organ. Rev.*, vol. 21, no. 2, pp. 299–315, 2021, doi: 10.1007/s11115-020-00492-3.
[4] H. Wahyuni, T. Purwaningsih, and N. Herlina, "Implementation of the Smart City Program to Improve Public Services in Semarang City," *J. NATAPRAJA Kaji. Ilmu Adm. Negara*, vol. 9, no. 2, pp. 131–147, 2021.
[5] D. A. Shafiq, N. Z. Jhanjhi, and A. Abdullah, "Load balancing techniques in cloud computing environment: A review," *J. King Saud Univ. - Comput. Inf. Sci.*, vol. 34, no. 7, pp. 3910–3933, 2022, doi: https://doi.org/10.1016/j.jksuci.2021.02.007.
[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] B. G. Malau, "Implementasi Load Balancing Mikrotik Jaringan Internet Di Pardamean Sibisa, Ajibata, Toba Samosir, Sumatra Utara," *JCS-TECH J. Comput. Sci. Technol.*, vol. 2, no. 1, pp. 20–29, 2022, doi: 10.54840/jcstech.v2i1.23.
[8] K. Azmi, S. Syamsul, and F. Razi, "Studi Penggunaan Dua ISP Dengan Load Balancing dan Failover Untuk Meningkatkan Kinerja Jaringan Berbasis Router Mikrotik," *J. TEKTRO*, vol. 06, no. 02, pp. 176–183, 2022.
[9] B. P. Mulla, C. R. Krishna, and R. K. Tickoo, *Load Balancing Algorithm for Efficient VM Allocation in Heterogeneous Cloud*. New York: SSRN, 2020.
[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] 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.
[12] 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.
[13] T. Octavriana, K. Joni, and A. F. Ibadillah, "Optimalisasi Jaringan Internet Dengan Load Balancing Pada High Traffic Network," *J. Tek. Inform.*, vol. 14, no. 1, pp. 28–39, 2021, doi: 10.15408/jti.v14i1.15018.
[14] S. Mohanty, *Evolutionary Approaches for Load Balancing in Cloud Computing*. Lagos: Bibi Incorporated, 2022.
[15] M. Mira, I. Ilhamsyah, and N. Mutiah, "Analisis Kualitas Layanan Sistem Informasi Smart City Gencil Kota Pontianak Menggunakan Indikator Smart City dan Model Webqual 4.0," *Coding J. Komput. dan Apl.*, vol. 07, no. 02, 2019.
[16] M. Jamil, S. Santosa, and M. Hamid, "Analisis Perbandingan Kinerja Load Balancing Menggunakan Metode PCC dan NTH," *J. PRODUKTIF*, vol. 7, no. 1, pp. 619–625, 2023.
[17] D. K. Hakim, D. Y. Yulianto, and A. Fauzan, "Pengujian Algoritma Load Balancing pada Web Server Menggunakan NGINX," *J. Ris. Sains dan Teknol.*, vol. 3, no. 2, pp. 85–92, 2019, doi: 10.30595/jrst.v3i2.5165.
[18] B. A. Alenizi, M. Humayun, and N. Z. Jhanjhi, "Security and Privacy Issues in Cloud Computing," *J. Phys. Conf. Ser.*, vol. 1979, no. 1, 2021, doi: 10.1088/1742-6596/1979/1/012038.
[19] P. Kumar and R. Kumar, "Issues and Challenges of Load Balancing Techniques in Cloud Computing: A Survey," *ACM Comput. Surv.*, vol. 51, no. 6, Feb. 2019, doi: 10.1145/3281010.
[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.
[21] D. Aribowo, "Cluster Server IPTV dengan Penjadwalan Algoritma Round Robin," vol. 1, no. 2, pp. 1–5, 2012.