Journal of Information System Research (JOSH)
Volume 3, No. 4, Juli 2022, pp 507−514
ISSN 2686-228X (media online)
https://ejurnal.seminar-id.com/index.php/josh/
DOI 10.47065/josh.v3i4.1792

Implementasi Metode *Load Balancing* Sebagai Upaya Meningkatkan Kinerja Server

Rini Nuraini

Fakultas Teknologi Komunikasi dan Informatika, Informatika, Universitas Nasional, Jakarta
Jl. Sawo Manila No.61, RT.14/RW.7, Pejaten Bar., Kec. Ps. Minggu, Kota Jakarta Selatan, DKI Jakarta, Indonesia
Email: [1]rini.nuraini@civitas.unas.ac.id
Email Penulis Korespondensi: rini.nuraini@civitas.unas.ac.id
Submitted: **01/07/2022**; Accepted: **23/07/2022**; Published: **31/07/2022**

Abstrak−Penggunaan internet ternyata menimbulkan persmasalahan yang harus dicari solusinya. Salah satunya adalah adanya beban server penyedia layanan yang terus meningkat. Kondisi tersebut dipicu oleh jumlah client yang terus bertambah dari waktu ke waktu. Bahkan beberapa situs melaporkan telah menerima secara simultan ratusan ribu permintaan koneksi dari sejumlah client. Dari permasalahn tersebut, maka diperlukan penelitian yang berfokus pada bagaimana merancang dan membangun sistem server yang mampu menangani peningkatan permintaan yang masuk supaya beban server dapat terurai. Tujuan dari penelitian ini adalah agar server penyedia layanan dapat meningkatkan layanannya terhadap client. Terdapat beberapa pendekatan untuk mengatasi permasalahan tersebut salah satunya yaitu dengan menerapkan metode load balancing. Sehingga dengan menerapkan load balancing, beban besar yang masuk akan didistribusikan ke masing-masing server penyedia layanan tersebut. Dalam pengujian yang telah dilakukan, skalabilitas sistem menjadi meningkat. Ini dibuktikan ketika sistem dengan load balancing diberiakan 10000 koneksi, hasil pengujian memiliki nilai rata-rata response time sebesar 44,42 ms. Sedangkan untuk sistem tanpa load balancing, hasil pengujian memiliki nilai rata-rata nilai response time sebesar 185,88 ms. Dari hasil pengujian terlihat bahwa nilai rata-rata response time dari sistem server dengan load balancing lebih kecil dibandingkan tanpa load balancing, maka kinerja layanan dari sistem dapat terus ditingkatkan dengan diterapkannya load balancing.

Kata Kunci: Response Time; Throughput; Load Balancing; Peningkatan Kinerja; Server

Abstract−The use of the internet turns out to cause problems that must be found a solution. One of them is the service provider's server load which continues to increase. This condition is triggered by the increasing number of clients from time to time. Some sites even reported receiving hundreds of thousands of connection requests simultaneously from a number of clients. From these problems, research is needed that focuses on how to design and build a server system that is able to handle the increase in incoming requests so that the server load can be unraveled. The purpose of this research is that the service provider's server can improve its service to the client. There are several approaches to overcome these problems, one of which is by applying the load balancing method. So that by implementing load balancing, the incoming large load will be distributed to each of the service provider's servers. In the tests that have been carried out, the scalability of the system has increased. This is proven when a system with load balancing is given 10000 connections, the test results have an average response time of 44.42 ms. As for the system without load balancing, the test results have an average response time value of 185.88 ms. From the test results, it can be seen that the average response time of the server system with load balancing is smaller than without load balancing, so the service performance of the system can be continuously improved by implementing load balancing.

Keywords: Response Time; Throughput; Load Balancing; Performance Improvement; Server

1. PENDAHULUAN

Pesatnya pertumbuhan internet akhir-akhir ini, berimbas terhadap meningkatnya jumlah pengguna yang terkoneksi dan mengakases sejumlah layanan yang tersedia secara *online*. Kondisi tesebut memicu meningkatnya beban server penyedia layanan seiring bertambahnya jumlah *client* dari waktu ke waktu [1]. Kondisi ini tentunya berdampak pada kebutuhan akan perangkat server yang besar dan kuat. Di samping itu, berbagai layanan berbasis aplikasi web dengan berbagai jenis saat ini, menjadikan akses terhadap layanan internet semakin meningkat. Ada beberapa jenis layanan aplikasi berbasis web yang tersedia, antara lain *E-Learning*, *E-Business*, *E-Government*, *E-Commerce*, *E-News* dan lain-lain [2][3]. Ragam dari perkembangan tersebut mendorong lahirnya teknologi terkini yang disebut dengan komputasi awan. Definisi komputasi awan disebutkan sebagai himpunan yang memiliki sumber daya komputasi dan jaringan selain itu juga memiliki model penyelesaian manajemen terhadap penyimpanan dan juga dukungan terhadap berbagai aplikasi yang berbasis virtual [4]. Yang memungkinkan aspek ketersediaan dapat dikondisikan menurut kebutuhan yang ada dengan pertimbangan seperti aspek ekonomi dan lainnya. Peranan utama yang dimiliki komputasi awan adalah dalam hal kemampuannya yang dinamis dalam mendukung infrastruktur IT, adanya jaminan terhadap kualitas layanan serta kemudahan konfigurasi terhadap layanan aplikasi [5][6]. Hadirnya teknologi komputasi awan sangat mempermudah dalam penyediaan infrastruktur. Sehingga tidak sedikit penelitian memanfaakan komputasi awan sebagai pilihan untuk meyediakan sumberdaya infrastruktur dalam membangun berbagai layanan yang berbasis web [7]. Sehingga untuk meningkatkan *performance* layanan dari sistem web server, sangat dibutuhkan sistem server yang kuat. Hal tersebut bertujuan untuk mengatasi permintaan tingginya akses terhadap server web tersebut. Sistem server yang kuat mendukung untuk ketersediaan terhadap layanannya ketika diperlukan. Model arsitektur yang didukungan oleh banyak server akan menjadi pilihan yang menarik untuk diterapkan. Dimana terdapat konsep mengenai *load balancing* yang dapat diterapkan. *Load balancing* merupakan salah satu teknik dalam jaringan yang dapat

Journal of Information System Research (JOSH)
Volume 3, No. 4, Juli 2022, pp 507−514
ISSN 2686-228X (media online)
https://ejurnal.seminar-id.com/index.php/josh/
DOI 10.47065/josh.v3i4.1792

digunakan untuk meningkatkan *availability* serta *performance* dari sebuah sistem server. Metode *load balancing* bekerja dengan cara membagikan beban koneksi yang diterima secara bersamaan ke berbagai server yang ada, langkah ini menjadikan beban pada sebuah server menjadi lebih kecil [8][9].

Tujuan dari penelitian yang dilakukan adalah untuk merancang, menguji serta mengevaluasi metode *load balancing* dalam meningkatkan *performance* layanan sistem server dalam menangani permintaan yang tinggi serta menjaga ketersedian layanan pada web server di dalam lingkungan komputasi awan. Dengan demikian parameter seperti *response time* dapat menjadi perhatian dalam mendukung sistem server yang baik. Disamping itu untuk mengatasi beban yang berlebih ketika jumlah akses meningkat, maka penerapan model klaster server dapat menjadi salah satu pilihan untuk menangani hal tersebut. Klaster server merupakan kumpulan dari server yang jumlahnya lebih dari satu. Dengan mengimplementasi model server klaster maka *availability* aplikasi maupun *reliability* sistem dapat ditingkatkan [10][11].

Load balancing merupakan teknik dalam jaringan komputer yang memiliki cara kerja dengan mendistribusikan permintaan yang masuk ke server kluster atau sejumlah komputer untuk mencapai pemanfaatan sumber daya yang ada secara optimal, memperkecil waktu respon, memperbesar nilai *throughput* serta menekan kemungkinan terjadinya *overload*. Server kluster merupakan kumpulan dari banyak perangkat komputer yang saling terhubung dan bekerjasama dalam banyak aspek. Sehingga server kluster dapat dimaknai sebagai kesatuan sistem yang menyatu. Secara umum kluster server bertujuan agar dapat menjamin *availability* serta meningkatkan *performance* dari sistem [12][13]. *Load balancing* memiliki dukungan yang multifungsi. Di samping menjadi penyeimbang beban, *load balancing* juga berperan sebagai pengatur dan pengalihan terhadap kepadatan jalur trafik. *Load balancing* juga dapat digunakan untuk menilai kesehatan dari sebuah server baik itu terhadap aplikasinya maupun konten yang terkandung, hal ini untuk meningkatkan layanan yang tersedia serta mempermudah pengelolaannya [14][15].

Ketika terjadi kondisi dimana jumlah *client* yang mengakses layanan server meningkat dan server tidak dapat mengatasinya, tentu ini akan menjadi masalah yang serius. Seringkali penyebabnya adalah beban berlebih yang diterima oleh suatu server tunggal sehingga layanan yang ada menjadi berhenti. Beberapa situs bahkan dilaporkan telah menerima ratusan ribu koneksi dari *client* secara simultan[16][17]. Berhentinya layanan tersebut merupakan dampak dari server penyedia layanan yang *down*. Saat ini, penggunaan server tunggal sudah tidak dapat direkomendasikan ketika jumlah *client* yang mengakases menjadi besar. Dari permasalahn yang ada, maka diperlukan penelitian yang berfokus pada bagaimana merancang sebuah sistem server yang mampu menangani terjadinya peningkatan layanan yang masuk agar beban dari server dapat terurai. Penelitian ini bertujuan untuk meningkatkan layanan server yang menyediakan layanan terhadap penggunannya. Salah satu solusi untuk mengatasi permasalahan tersebut adalah dengan menerapkan metode *load balancing*. Akhirnya dengan menerapkan metode *load balancing*, beban besar yang masuk akan didistribusikan ke masing-masing server penyedia layanan tersebut.

2. METODOLOGI PENELITIAN

2.1 Tahapan Penelitian

Adapun berbagai tahapan yang harus dilakukan dalam melaksanakan penelitian ini adalah seperti yang tunjukkan pada Gambar 1. Pada pengujian yang akan dilakukan, metode *load balancing* digunakan untuk mendistribusikan sejumlah koneksi yang masuk ke masing-masing web server, selanjutnya akan dilakukan pengukuran peningkatan kinerja dengan cara melakukan pengujian terhadap kondisi sebelum dan setelah diterapkannya metode *load balancing*. Pada tahap awal penelitian terlebih dahulu melakukan tinjauan pustaka. Tahap berikutnya adalah membangun server web. Setelah server web selesai dibangun kemudian dilanjutkan membangun server *load balancing*. Dalam mengukur kinerjanya, sejumlah koneksi akan diberikan secara bertahap kepada web server. Koneksi akan diarahkan kepada server web, baik itu server tunggal maupun dalam kumpulan server web yang sudah terintegrasi dengan server *load balancing*. Kemudaian untuk tahap selanjutnya dilakukan evaluasi terhadap teknik *load balancing* yang telah dibangun menggunakan berbagai skenario pengujian yang sudah ditentukan sebelumnya. Hasil yang diperoleh dari pengujian akan dilakukan evaluasi guna melihat sejauh mana peran *load balancing* terhadap kinerja sistem.

Gambar 1. Metodologi Penelitian

Journal of Information System Research (JOSH)
Volume 3, No. 4, Juli 2022, pp 507−514
ISSN 2686-228X (media online)
https://ejurnal.seminar-id.com/index.php/josh/
DOI 10.47065/josh.v3i4.1792

Untuk melaksanakan penelitian ini, dibutuhkan beragam informasi dan keterangan yang komprehensif. Terdapat sejumlah tahapan yang dilaksanakan dalam proses pengumpulan data dalam penelitian ini, yaitu:

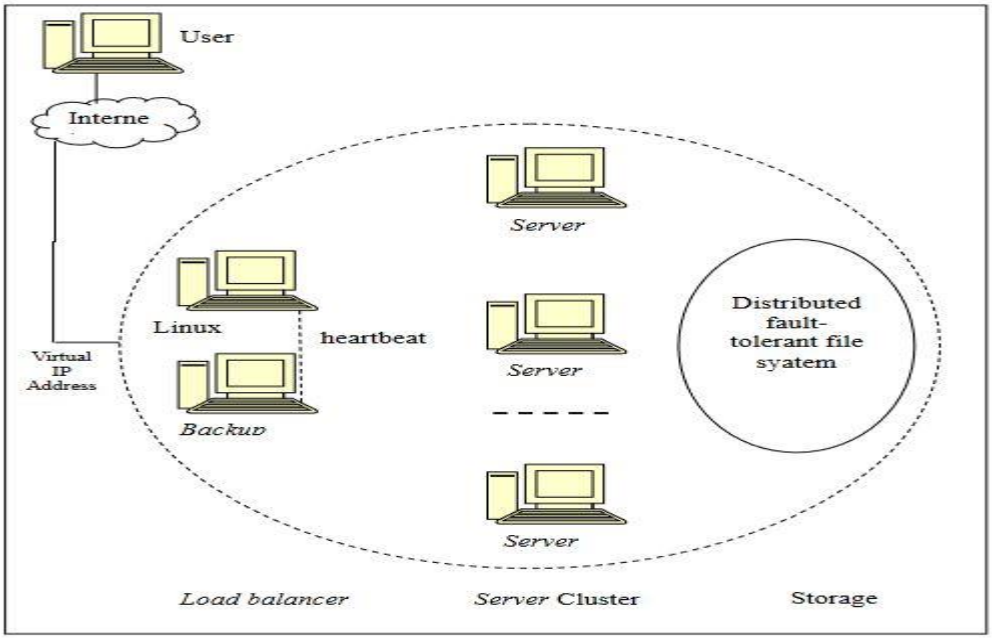
1) Obervasi dan tinjauan literatur
   Pada bagian ini pengumpulan informasi dari berbagai macam sumber bahan tulisan seperti artikel jurnal, prosiding, buku serta dokumen pendukung lainnya [18][19]. Maka, pada penelitian ini dilakukan pengumpulan data berupa sumber-sumber reverensi yang berhubungan dengan penelitian.
2) Implementasi server web pada mesin virtual
   Konfigurasi server web pada lingkungan mesin virtual.
3) Implementasi *load balancing* pada server di mesin virtual
   Konfigurasi *load balancing* pada server pada lingkungan mesin virtual. Dimana nantinya terdapat 3 buah server dan 1 buah *client* yang akan dikonfigurasi dan digunakan untuk pengujiannya.
4) Menguji dan mengevaluasi kinerja server
   Bagian ini dilakukan pengujian agar kinerja server dapat diketahui baik sebelum dan sesudah *load balancing* berhasil diimplementasikan. Setalah itu baru dilaksanakan evaluasi terhadap hasil pengujian yang didapat.
5) Dokumentasi terhadap hasil pengujian
   Dokumentasi hasil pengujian bertujuan sebagai bahan informasi yang bisa dimanfaatkan dan digunakan untuk penelitian selanjutnya.

2.2 Skenario Pengujian *Load balancing*

Dalam pengujian sistem, ada sejumlah skenatrio yang akan dijalankan. Skenario pengujian ini dilakukan terhadap server *load balancing* yang telah dibangun maupun terhadap server tanpa *load balancing*. Hal tersebut dilakukan untuk menguji kemampuan dari sistem *load balancing*, yaitu dengan membangkitkan koneksi dengan jumlah koneksi yang tidak sama, yaitu antara 5000/500, 6000/600, 7000/700, 8000/800, 9000/900 dan 10000/1000 koneksi pada waktu tertentu untuk mendapatkan nilai parameter dari *response time*.

2.3 Arsitektur Sistem Server Load balancing

Arsitektur dari server *load balancing* ini memiliki tiga bagian sebagaimana ditunjukkan pada Gambar 2, dimana terdapat sejumalah server yang terdiri dari server *load balancer*, server aplikasi 1 sampai ke-n serta server database.



Gambar 2. Arsitektur *Server Load Balancer*

Pada bagian yang pertama terdapat server *load balancer*. Server *load balancer* memiliki peran dalam mendistribusikan beban koneksi yang masuk ke masing-masing server aplikasi. Server *load balancing* ini dikonfigurasi dengan alamat IP yang dapat terkoneksi dengan masing-masing server aplikasi dan juga komputer *client*. Selanjutnya untuk bagian ke-2 merupakan bagian server aplikasi yang terdiri dari sejumlah server. Namun pada penelitian ini digunakan 2 buah server yang difungsikan sebagai server web. Server web 1 dikonfigurasi dengan alamat IP 192.168.109.179 sedangkan untuk server web 2 dikonfigurasi alamat IP 192.168.109.178. Kedua server menjalankan program aplikasi web yang nantinya akan diakses oleh *client*. Sedangkan untuk bagian yang ketiga bisa ditempatkan sebuah server yang berfungsi sebagai storage atau penyimpanan database yang terdiri dari sebuah server database. Dalam penelitian ini database langsung ditempatkan pada masing-masing server web

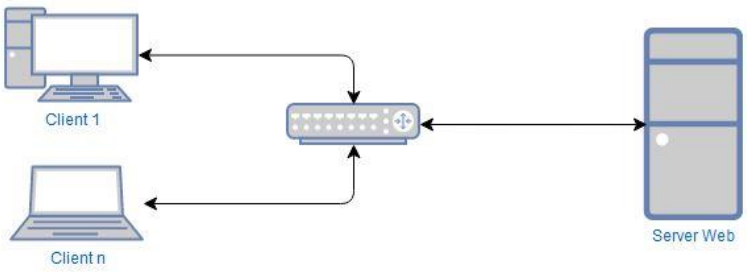
Journal of Information System Research (JOSH)
Volume 3, No. 4, Juli 2022, pp 507−514
ISSN 2686-228X (media online)
https://ejurnal.seminar-id.com/index.php/josh/
DOI 10.47065/josh.v3i4.1792

melaikan tidak dipisah. Sistem ini di implementasikan dan diujikan dalam lingkungan virtual mesin. Berikut merupakan spesifikasi server yang digunakan dalam penelitian, antara lain:

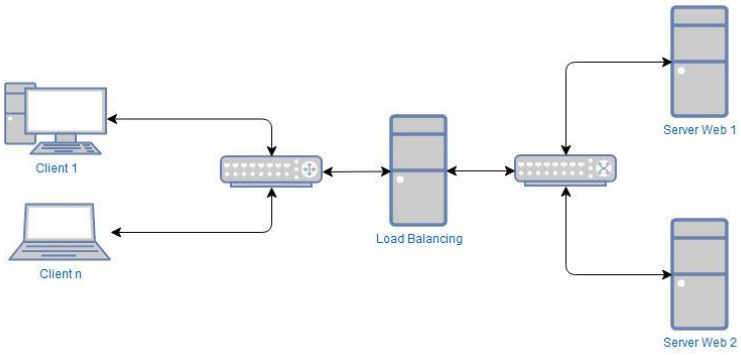
1. Satu unit komputer utama yang memiliki spesifikasi Proc. AMD Ryzen 5 with Radeon Vega Mobile Gfx (8 CPUs) - 2,0 GHz, RAM 8 GB, SSD 250 GB, dengan Windows 10.
2. Satu unit vitual mesin untuk server *load balancing* yang memiliki spesifikasi Proc. AMD Ryzen 5 with Radeon Vega Mobile Gfx (1 CPUs) - 2,0 GHz, RAM 1 GB, SSD 20 GB, dengan OS Linux Ubuntu Server
3. Dua unit virtual mesin untuk server application yang memiliki spesifikasi Proc. AMD Ryzen 5 with Radeon Vega Mobile Gfx (1 CPUs) - 2,0 GHz, RAM 1 GB, SSD 20 GB, dengan OS Linux Ubuntu Server.
4. Satu unit virtual mesin untuk client yang memiliki spesifikasi Proc. AMD Ryzen 5 with Radeon Vega Mobile Gfx (1 CPUs) - 2,0 GHz, RAM 1 GB, SSD 20 GB, dengan OS Linux Ubuntu Server.

3. HASIL DAN PEMBAHASAN

Pengujian terhadap sejumlah parameter pada server *load balancing* memiliki tujuan untuk mengukur kinerja dari sistem server dalam menerima banyaknya *request* masuk dari *client* dalam rentang waktu tertentu. Pada proses pengujian sistem server, parameter yang diukur adalah parameter *response time*. Pengukuran parameter tersebut dilakukan baik sebelum dan sesudah *load balancing* berhasil dibangun. Dalam membangun sistem *load balancing*, digunakan virtual mesin yang berada pada komputer utama untuk membangun infrastruktur server-nya. Pada penelitian ini, terdapat dua rancangan yang akan dievaluasi dari arsitektur server yang ada. Model pertama merupakan rancangan dari arsitektur server tunggal. Ini merupakan arsitektur yang belum menerapkan metode *load balancing*. Gambar 3 merupakan arsitektur tanpa *load balancing*. Pada arsitektur ini, hanya terdapat satu server untuk bertugas untuk melayani permintaan yang datang dari pengguna. Jadi sebanyak apapun permintaan yang masuk, maka permintaan akan di arahkan ke server tersebut. Kemudian untuk arsitektur yang ke dua telah menerapkan sejumlah server yang berfungsi untuk melayani permintaan yang masuk seperti yang ditunjukkan pada Gambar 4. Arsitektur pada Gambar 4 telah mengusung konsep mengenai metode *load balancing* yang akan dibangun. Dimana rancangan arsitektur ini terdiri dari satu server *load balancing* dan dua buah server web. Dalam prosesnya, permintaan yang masuk dari *client* tidak akan dilayani server tunggal, melainkan sudah dilayani oleh 2 server atau lebih secara bersamaan.



Gambar 3. Model arsitektur dengan *server* tunggal



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

Pada Gambar 4 merupakan rancangan sistem server yang terdiri dari server yang bertindak sebagai *load balancing*, server aplikasi web 1, server aplikasi web 2 dan *client* yang terletak pada satu jaringan yang terhubung melalui *switch*. Dengan demikian *client* dapat mengakses melalui jaringan tersebut untuk memperoleh layanan

Journal of Information System Research (JOSH)
Volume 3, No. 4, Juli 2022, pp 507−514
ISSN 2686-228X (media online)
https://ejurnal.seminar-id.com/index.php/josh/
DOI 10.47065/josh.v3i4.1792

dari server web 1 maupun server web 2. Untuk alamat IP yang terpasang pada perangkat jaringan dapat dilihat pada tabel 1.

Tabel 1. Nama Perangkat Server dan IP Address

Nama Perangkat	IP Address
Client	192.168.109.182
Server Web 1	192.168.109.179
Server Web 2	192.168.109.178
Server *Load balancing*	192.168.109.181

Pada tabel 1, berisikan informasi mengenai alamat IP yang terpasang dimasing-masing perangkat. Dapat dilihat bahwa alamat dari IP dari server *load balancing* adalah 192.168.109.181. Server *load balancing* memiliki fungsi untuk mendistribusikan layanan yang masuk dari *client* ke web server. Ketika *client* mengakses web server untuk mendapatkan layanan, maka secara otomatis akan diarahkan untuk memasuki server *load balancing*. Setelah itu *request* tersebut akan diteruskan ke masing-masing server web yang terdapat di bagian belakang dari server *load balancing*. Jadi pada skenario ini, *client* tidak mengetahui server web mana yang bertanggung jawab dan bertugas untuk melayani permintaan yang masuk. Selanjutnya, permintaan layanan yang masuk akan diteruskan server *load balancing* kepada server web yang telah terdaftar pada tabel penugasan dari server *load balancing*. Pengujian yang dilakukan pada penelitian ini yaitu dengan mengamati nilai *response time* yang telah terukur. Tujuannya yaitu untuk mengukur kinerja layanan yang akan diberikan oleh server web ketika server *load balancing* sudah diterapkan. Dalam proses ujinya, *client* akan memulai koneksi secara simultan menggunakan *software benchmark* httperf dalam memperoleh variabel nilai *response time*. Untuk deskripsi jelasnya mengenai hasil pengujian dapat dilihat pada Tabel 2 dan Tabel 3 berikut.

Tabel 2. Hasil pengujian *response time (ms)* Server dengan *load balancing*

No	***Respontime (ms)* Server dengan *Load balancing***						
	Tingkat Koneksi	**Uji ke-1**	**Uji ke-2**	**Uji ke-3**	**Uji ke- 4**	**Uji ke-5**	**Hasil Rata-rata Pengujian**
1	5000/500	2.9	3	3	3.7	3	3.12
2	6000/600	4.3	3.3	3	4.6	3.4	3.72
3	7000/700	4	3.9	4.3	4.1	4.5	4.16
4	8000/800	7	9.9	7.1	8.1	10	8.42
5	9000/900	17	15	16.5	17.9	15.7	16.42
6	10000/1000	29.4	41.2	54.7	57.2	39.6	44.42

Tabel 3. Hasil pengujian *response time (ms)* Server tanpa *load balancing*

No	***Respontime (ms)* Server tanpa *Load balancing***						
	Tingkat Koneksi	**Uji ke-1**	**Uji ke-2**	**Uji ke-3**	**Uji ke- 4**	**Uji ke-5**	**Hasil Rata-rata Pengujian**
1	5000/500	1.7	1.6	1.6	1.7	2.6	1.84
2	6000/600	3.5	2.1	1.7	3.2	1.7	2.44
3	7000/700	2.9	2.4	2.4	2.4	2.6	2.54
4	8000/800	5.8	6.9	5.8	5.9	7.5	6.38
5	9000/900	14.1	16.2	22.1	13.4	15.8	16.32
6	10000/1000	121.7	221.1	232.9	195.5	158.2	185.88

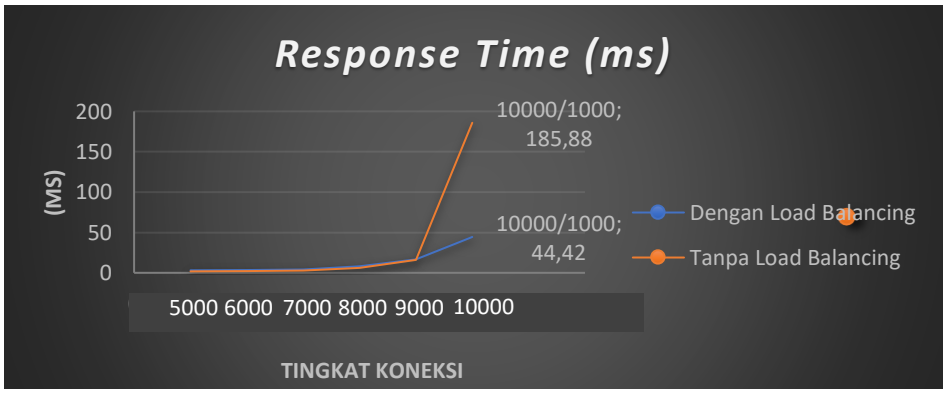
Tabel 2 dan tabel 3 berisikan informasi hasil pengujian server dengan *load balancing* dengan membuat sejumlah koneksi yang dilakukan oleh *client* melalui *software benchmark* httperf yang diarahkan pada server web. Permintaan koneksi oleh *client* ke server web dilakukan bertahap. Pada skenario awal dimulai dengan membuat koneksi untuk 5000 *request* untuk 500 koneksi/*second*. Kemudian dilanjutkan 6000 *request* untuk 600 koneksi/*second*. Berikutnya dengan 7000 *request* dengan 700 koneksi/*second*. Dilanjutkan dengan 8000 *request* dengan 800 koneksi/*second*, 9000 *request* dengan 900 koneksi/*second* dan terakhir dengan 10000 *request* dengan 1000 koneksi/*second* dan dilaksanakan secara bertahap terhadap server web. Dari hasil dari pengujian, terdapat nilai yang berbeda untuk setiap skenario pengujiannya. Kondisi ini muncul akibat adanya rentang waktu yang lebih panjang untuk setiap skenario pengujiannya. Sehingga menimbulkan antrian yang berbeda antar skenario untuk setiap permintaan yang harus diproses. Peristiwa tersebut memberikan dampak pada nilai *response time* yang akan mengalami peningkatan seiring meningkatnya *request* terhadap layanan sever web. Berdasarkan nilai *response time* yang telah didapatkan dari hasil pengujian, maka penerapan kinerja dari perbandingan kedua rancangan arsitektur dapat diketahui. Kemudian, untuk hasil rata-rata pengujian *response time* disajikan pada Tabel 4.

Journal of Information System Research (JOSH)
Volume 3, No. 4, Juli 2022, pp 507−514
ISSN 2686-228X (media online)
https://ejurnal.seminar-id.com/index.php/josh/
DOI 10.47065/josh.v3i4.1792

Tabel 4. Hasil rata-rata pengujian *response time (ms)* Server pada kedua arsitektur

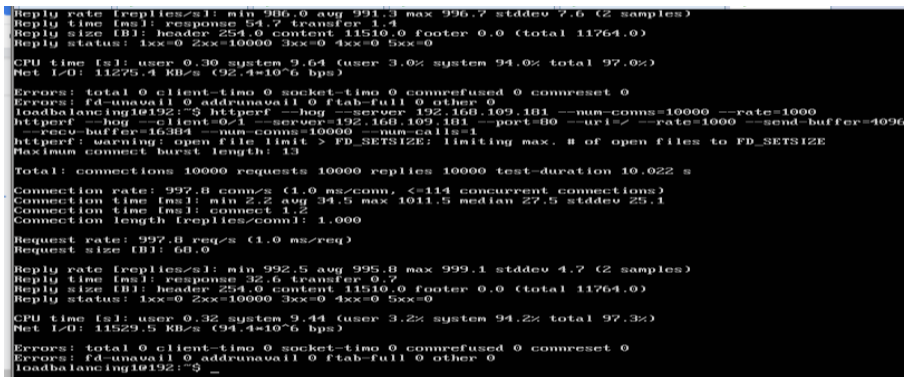
Tingkat Koneksi	*Response time (ms)*	
	Dengan *Load balancing*	Tanpa *Load balancing*
5000/500	3.12	1.84
6000/600	3.72	2.44
7000/700	4.16	2.54
8000/800	8.42	6.38
9000/900	16.42	16.32
10000/1000	44.42	185.88

Pada tabel 4 berisikan informasi mengenai perbandingan hasil rata-rata pengujian terhadap nilai *respontime*. Dimana terlihat bahwa untuk tingkat koneksi yang sangat besar yaitu 10000 request dengan 1000 koneksi/second, server dengan *load balacing* masih mampu mengani dengan baik dilihat dari nilai *respontime* yang jauh lebih kecil dibanding dengan tanpa *load balacing*.



Gambar 5. Hasil nilai *response time* antara model satu server dan banyak server

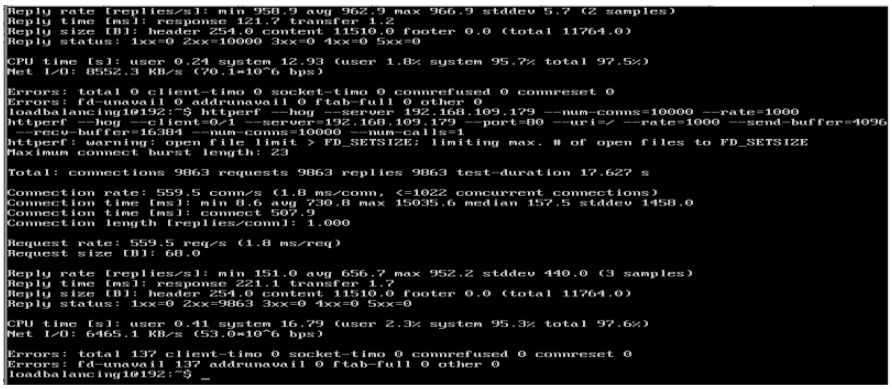
Pada Gambar 5 merupakan hasil perbandingan yang dibuat dalam bentuk grafik antara model dengan *load balancing* dengan tanpa *load balancing*. Kondisi tersebut terlihat dari hasil pengujian berdasarkan pada waktu respon yang diperoleh. Nilai rata-rata pada pengujian kedua model arsitektur baik dengan *load balancing* dengan tanpa *load balancing* dapat dilihat pada tabel 4. Dari hasil pengujian yang dilakukan, maka dengan implementasi *load balancing* nilai *response time* didapat jauh lebih kecil untuk jumlah koneksi yang besar dibandingkan tanpa *load balancing*. Pengujian dilakukan dengan rentang koneksi dan waktu yang berbeda. Pada uji koneksi dengan nilai *request* sebanyak 5000 *request* dengan 500 koneksi/*second*, 6000 *request* dengan 600 koneksi/*second*, 7000 *request* dengan 700 koneksi/*second*, 8000 *request* dengan 800 koneksi/*second*, dan 9000 *request* dengan 900 koneksi/*second* server tunggal terlihat sedikit lebih unggul. Kondisi tersebut dikarenakan jumlah koneksi yang masih dapat ditangani langsung oleh server web tunggal. Namun hal berbeda terjadi pada rentang waktu pengujian 10000 *request* dengan 1000 koneksi/*second*. Server tunggal mengalami peningkatan nilai *response time* yang sangat tinggi. Sangat berbeda jika dibandingkan dengan sistem yang telah menerapkan *load balancing*. Kondisi tersebut memperlihatkan bahwa nilai *response time* yang di-*support* oleh penerapan *load balancing* lebih baik dibandingkan hanya menggunakan server tunggal untuk melayani koneksi yang tinggi. Dimana permintaan yang masuk masih dapat dilayani seperti yang ditunjukkan pada Gambar 6.



Gambar 6. Hasil pengujian koneksi 10000/1000 dengan *load balancing*

Berdasarkan Gambar 6, maka penerapan *load balancing* sangat mendukung untuk mencapai nilai QoS yang baik. Fenomena lain juga terjadi saat pengujian nilai 10000 *request* dengan 1000 koneksi/*second*. Pengujian dilanjutkan dengan uji 2 dan uji 3 dengan hasil uji seperti pada Gambar 7 dan Gambar 8 berikut ini.

Journal of Information System Research (JOSH)
Volume 3, No. 4, Juli 2022, pp 507−514
ISSN 2686-228X (media online)
https://ejurnal.seminar-id.com/index.php/josh/
DOI 10.47065/josh.v3i4.1792



Gambar 7. Hasil pengujian koneksi 10000/1000 tanpa *load balancing* pada Uji 2

Gambar 8. Hasil pengujian koneksi 10000/1000 tanpa *load balancing* pada Uji 3

Berdasarkan pada Gambar 7 dan Gambar 8, terlihat bahwa uji koneksi tersebut server sudah tidak dapat melayani sepenuhnya terhadap permintaan yang masuk. Kondisi tersebut terlihat dari nilai total *errors* yang terekam saat dilakukan pengujian. Maka, hal ini menunjukkan bahwa dengan penerapan *load balancing* dapat meningkatkan kinerja server.

4. KESIMPULAN

Penelitian ini bertujuan untuk melihat dampak dari implementasi *load balancing* pada sebuah sistem server yang berperan untuk mendistribusikan permintaan ke masing-masing web server. Hasil pengujian terhadap implementasi dari *load balancing* didapatkan nilai *response time* yang lebih kecil untuk pengujian koneksi 10000/1000 yaitu 44,42 ms dibandingkan tanpa *load balancing* dengan nilai sebesar 185.88 ms. Dari sini terlihat bahwa pembagian beban koneksi yang masuk dapat didistribusikan secara merata pada sejumlah server web oleh server *load balancing*. Hal ini tentunya sangat berpengaruh terhadap *response time* yang diberikan server dalam melayani permintaan. Untuk penelitian yang akan datang server *load balancing* harus memiliki cadangan dengan menambah jumlah server *load balancing* yang dioperasikan. Upaya ini untuk mendukung model *fault tolerance* yang sangat dibutuhkan untuk meningkatkan ketersediaan layanan dari sistem server.

REFERENCES

[1] I. Sujarwo, D. Desmulyati, and I. Budiawan, "Implementasi Load Balancing Menggunakan Metode PCC (Per Connection Clasifier) Di Universitas Krisnadwipayana," *JITK (Jurnal Ilmu Pengetah. dan Teknol. Komputer)*, vol. 5, no. 2, pp. 171–176, 2020, doi: 10.33480/jitk.v5i2.1184.

[2] S. D. Riskiono and D. Pasha, "Analisis Metode Load Balancing Dalam Meningkatkan Kinerja Website E-Learning," *J. Teknoinfo*, vol. 14, no. 1, p. 22, 2020, doi: 10.33365/jti.v14i1.466.

[3] I. Arnomo, "Simulasi Pengamanan Database Web Server Repository Institusi Melalui Jaringan LAN Menggunakan Remote Access," *J. Sist. Informasi, Teknol. Inform. dan Komput.*, vol. 9, no. 1, pp. 64–71, 2018.

[4] I. Ahmad, E. Suwarni, R. I. Borman, A. Asmawati, F. Rossi, and Y. Jusman, "Implementation of RESTful API Web Services Architecture in Takeaway Application Development," in *International Conference on Electronic and Electrical Engineering and Intelligent System (ICE3IS)*, 2022, pp. 132–137.

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

[6] S. Suresh and S. Sakthivel, "A novel performance constrained power management framework for cloud computing using an adaptive node scaling approach," *Comput. Electr. Eng.*, vol. 60, pp. 30–44, 2017, doi: 10.1016/j.compeleceng.2017.04.018.

[7] R. I. Borman, K. Syahputra, J. Jupriyadi, and P. Prasetyawan, "Implementasi Internet of Things pada Aplikasi Monitoring

Journal of Information System Research (JOSH)
Volume 3, No. 4, Juli 2022, pp 507−514
ISSN 2686-228X (media online)
https://ejurnal.seminar-id.com/index.php/josh/
DOI 10.47065/josh.v3i4.1792

Kereta Api dengan Geolocation Information System," in *Seminar Nasional Teknik Elektro 2018*, 2018, pp. 322–327.
[8] F. Apriliansyah, I. Fitri, and A. Iskandar, "Implementasi Load Balancing Pada Web Server Menggunakan Nginx," *J. Teknol. dan Manaj. Inform.*, vol. 6, no. 1, 2020, doi: 10.3997/2214-4609.201801770.
[9] H. Ren, Y. Lan, and C. Yin, "The load balancing algorithm in cloud computing environment," *Proc. 2nd Int. Conf. Comput. Sci. Netw. Technol. ICCSNT 2012*, pp. 925–928, 2012, doi: 10.1109/ICCSNT.2012.6526078.
[10] S. D. Riskiono, S. Sulistyo, and T. B. Adji, "Evaluasi Metode Load Balancing Menggunakan Haproxy Server Chat Social Network," pp. 635–639, 2016.
[11] S. D. Riskiono, S. Sulistyo, and T. B. Adji, "Kinerja Metode Load Balancing dan Fault Tolerance Pada Server Aplikasi Chat," *Pros. Semin. Nas. ReTII*, 2017.
[12] U. Haluoleo, K. Bumi, and T. Anduonohu, "Peningkatan Kinerja Siakad Menggunakan Metode Load Balancing dan Fault Tolerance Di Jaringan Kampus Universitas Halu Oleo," vol. 10, no. 1, pp. 11–22, 2016.
[13] 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, [Online]. Available: https://journal.akprind.ac.id/index.php/jarkom/article/view/3670/2671
[14] M. Kumar and S. C. Sharma, "Dynamic load balancing algorithm for balancing the workload among virtual machine in cloud computing," *Procedia Comput. Sci.*, vol. 115, pp. 322–329, 2017, doi: 10.1016/j.procs.2017.09.141.
[15] N. Fauzi, W. Yahya, and A. Bhawiyuga, "Implementasi Load Balancing Pada Server Dengan Menggunakan Algoritme Least Traffic Pada Software-Defined Network," *J. Pengemb. Teknol. Inf. dan Ilmu Komput. Univ. Brawijaya*, vol. 2, no. 9, pp. 3134–3141, 2018, [Online]. Available: http://j-ptiik.ub.ac.id/index.php/j-ptiik/article/view/2565%0Ahttp://j-ptiik.ub.ac.id/index.php/j-ptiik/article/download/2565/945
[16] 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.
[17] 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.
[18] A. Amarudin and S. D. Riskiono, "Analisis Dan Desain Jalur Transmisi Jaringan Alternatif Menggunakan Virtual Private Network (Vpn)," *J. Teknoinfo*, vol. 13, no. 2, p. 100, 2019, doi: 10.33365/jti.v13i2.309.
[19] I. Ahmad, A. T. Prastowo, E. Suwarni, and R. I. Borman, "Pengembangan Aplikasi Online Delivery Sebagai Upaya Untuk Membantu Peningkatan Pendapatan," *JMM (Jurnal Masy. Mandiri)*, vol. 5, no. 6, pp. 4–12, 2021.