Perancangan Dan Implementasi Web Dengan Load Balancer Untuk Membantu Manajemen Trans Metro Bandung

Ilyas Adiyasa
Fakultas Teknik Elektro
Universitas Telkom
Bandung, Indonesia
adiyasailyas@student.telkomuniversity.ac.id

Rendy Munadi
Fakultas Teknik Elektro
Universitas Telkom
Bandung, Indonesia
rendymunadi@telkomuniversity.ac.id

Sussi
Fakultas Teknik Elektro
Universitas Telkom
Bandung, Indonesia
sussiss@telkomuniversity.ac.id

Abstrak—Pada penelitian kali ini, peneliti menemukan masalah dari Trans Metro Bandung (TMB) masih melakukan pengawasan secara manual untuk protokol kesehatan penumpang, dan perhitungan penumpang pada tiap harinya belum tersedia. Dengan masalah yang ditemukan, peneliti memberikan alternatif berupa pembuatan suatu *website* yang difungsikan sebagai sarana *monitoring* untuk armada bus. Hasil yang didapatkan pada pengujian terhadap rancangan yang telah dilakukan implementasi adalah *load balancer* dengan algoritma *Round Robin* dapat meringankan beban dari *server website* yang sebelumnya tidak menggunkan *load balancer.* Hal tersebut dapat disimpulkan oleh penulis dikarenakan *server* yang menggunakan *load balancer* selalu unggul pada perbandingan *error request, throughput,* dan *delay.* Untuk perbandingan *error request, throughput (received), throughput (sent), throughput (request per second),* dan *delay* pada percobaan *request* sejumlah 800 antara *server* yang menggunakan *load balancer* dan yang tidak menggunakan *load balancer* berturut-turut sebesar. 0% dengan 28.88%, 82.426 *KB/sec* dengan 268.538 *KB/sec*, 110.418 *KB/sec* dengan 66.396 *KB/sec*, 357.8175 *req/sec* dengan 316.7676 *req/sec*, dan 103 ms dengan 682.4 ms. Kemudian untuk pengujian jumlah pendapatan TMB, hanya saja masih kurang akurat dikarenakan untuk perhiungan pengguna *masker* dan tanpa *masker* tidak akurat. Pengujian terakhir mengenai kemudahan akses *website* yang telah dibuat mendapatkan kesimpulan mudah digunakan dari beberapa *feedback* yang ada.

Kata kunci—*Bus Rapid Transit, connections, delay, error socket, feedback, load balancer, monitoring, throughput,* Trans Metro Bandung, *website*

I. PENDAHULUAN

Pemerintah kota Bandung merencanakan program angkutan umum cepat masal atau Bus Rapid Transit (BRT). BRT adalah sistem angkutan berbasis bis berkualitas tinggi, yang bergerak dengan cepat, nyaman, dan efektif pada suatu infrastruktur jalur jalan yang terpisah, mempunyai karakteristik operasional yang cepat dengan frekuensi tertentu, serta mempunyai sistem pemasaran dan layanan pelanggan yang prima [1]. Hal tersebut menyebabkan angkutan umum pada kota Bandung termasuk TMB tidak memiliki jalur sendiri yang akan mengakibatkan angkutan umum ikut terjebak dalam kemacetan lalu lintas kota Bandung, sehingga waktu perjalanan angkutan umum tergantung oleh lancer tidaknya lalu lintas yang dilewati.

Hubungan antara kemacetan dan angkutan umum kota Bandung adalah keterbatasan dalam memberikan tingkat kepercayaan masyarakat untuk menggunakan layanan transportasi umum dikarenakan tidak adanya estimasi waktu kedatangan atau lama perjalanan yang dijanjikan oleh pihak angkutan umum termasuk TMB. Dari beberapa masalah serta referensi yang didapatkan oleh penulis, penulis memutuskan untuk menyelesaikan masalah yang ada dengan membuat website untuk pihak TMB. Website tersebut diharapkan akan membantu admin TMB untuk melakukan monitoring serta pengguna biasa untuk melihat informasi yang ada dan melakukan feedback terhadap TMB.

Pembuatan website untuk TMB akan menyelesaikan masalah seperti yang telah disebutkan di atas, namun terdapat masalah baru bila dilakukannya digitalisasi dalam bentuk website, dengan data penumpang yang kita peroleh, pengguna dari website yang akan dibuat juga tidak sedikit, masalah yang akan dihadapi yaitu terdapat kemungkinan website yang telah dibuat tidak dapat melayani semua permintaan dari pengguna biasa maupun admin. Dengan masalah tersebut, dapat diselesaikan dengan cara memberikan fitur yang disebut load balance.

Pada penelitian Tugas Akhir ini akan diberi nama “Smart Metro” yang terdiri dari website yang dirancang memiliki dua buah jenis pengguna, yaitu admin dari TMB dan pengguna pada umumnya. Website ini dirancang memiliki fitur login atau ‘masuk’ untuk dapat mengakses website tersebut. Untuk login website akan terkoneksi melalui database yang bersifat tidak relasional atau berbasis object menggunakan platform dari Firebase yaitu Firestore database. Kemudian untuk fitur penampil jumlah penumpang yang akan dijadikan perhitungan revenue dari TMB akan diproses melalui hasil image processing dari deteksi masker akan langsung dikirim ke Firestore database.

II. KAJIAN TEORI

A. Website

Website yaitu kumpulan beberapa halaman yang memuat informasi dalam bentuk teks, citra dapat berbentuk gambar, video, hingga animasi yang digitalkan dan dapat bersifat statis maupun dinamis agar pengguna dapat mengakses melalui perangkat lunak yang terkoneksi ke internet [1]. Pada proyek ini, *Website* dirancang menggunakan bahasa pemrograman *JavaScript* untuk bagian *front end* dan *backend.* Pada bagian *front end* menggunakan *framework* dari bahasa pemrograman *JavaScript* yaitu *React.JS* yang akan dihias dengan *framework SASS*. Kemudian untuk bagian

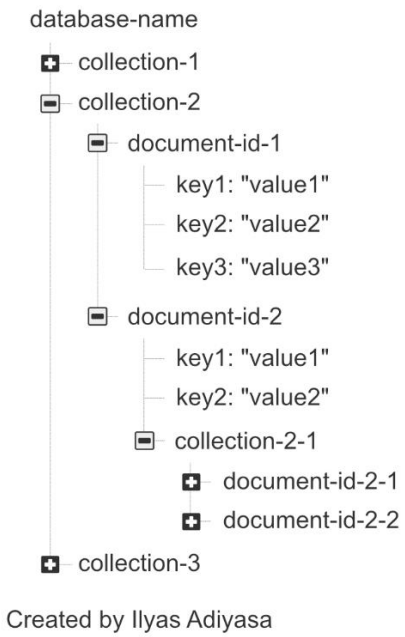
back end menggunakan *platform Cloud Firebase* dan akan menggunakan *database Cloud Firestore*.

B. ReactJs

React.Js pada dasarnya dirancang untuk mengatasi masalah kinerja di aplikasi web. React menggunakan virtual DOM yang memutuskan suatu komponen harus dimuat ulang atau tidak berdasarkan *current state* komponen saat ini dan atau terjadi perubahan. Hal tersebut mencegah aplikasi dari melakukan *render* ulang yang tidak diperlukan. *State* dan *props* dari komponen adalah dua parameter yang menentukan kapan komponen harus dilakukan *render* ulang di aplikasi. Bila terjadi perubahan atau ketika *parent* dari komponen memberikan *props* baru kepada *child*, *React DOM* membandingkan nilai baru dengan nilai sebelumnya nilai yang disimpan dan hanya dirender ulang jika terdapat perbedaan antara dua *states* yang berbeda [2].

C. Cloud Firebase (Firestore Database)

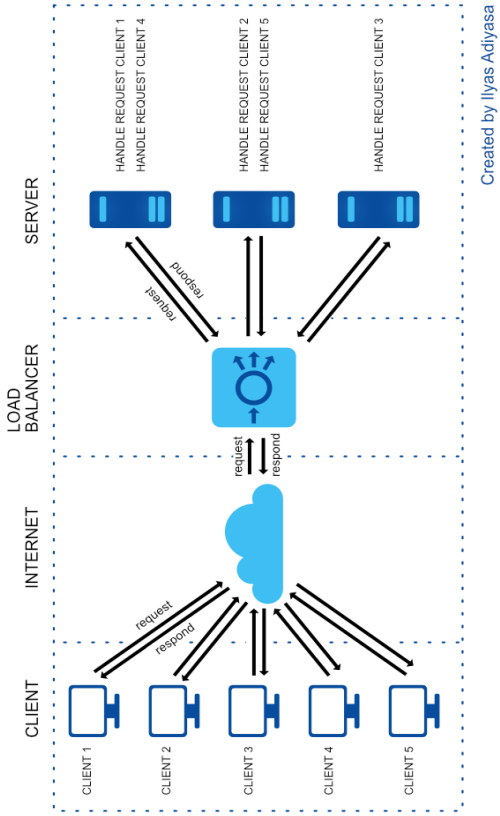
Firebase merupakan kumpulan dari beberapa layanan dari Google yang menjadi satu layanan pada *cloud*. Layanan dari *Firebase* terdiri dari *user authentication*, *real-time database*, *firestore database*, *storage*, dan lain sebagainya [3]**.** *Firestore Database* ini termasuk jenis *database NoSQL* yang dapat memberikan layanan untuk terus menerus melakukan penyimpanan data dan melakukan pembaruan [4]. Seperti yang dijelaskan pada **Gambar 2.3**, *database* ini memiliki sistem berbentuk seperti wadah dan memiliki banyak *document* di dalam wadah tersebut. Di dalam *document* pasti terdapat coretan atau tulisan penting. Jadi pada *firestore* wadah tersebut disebut juga *collection* dan *document* tersebut disebut juga *document*, lalu untuk isi atau coretan dari *document* tersebut kerap disebut *key and value.* Tidak hanya itu, *firestore* mengizinkan pengembang *database* untuk membuat *collection* di dalam *collection.*



Struktur Firestore database (C)

D. Load Balancing

Load balancing adalah metode untuk membagi beban trafik pada jumlah jalur koneksi yang ditentukan, dapat berjumlah 2 jalur maupun lebih. Metode ini bertujuan agar trafik koneksi dapat berjalan optimal. Metode *Load balancing* dilakuakn ketika server memiliki pengguna yang telah melebihi kemampuan penerimaan *request*. Salah satu alat yang digunakan untuk melakukan *load balancing* adalah *Nginx* [5]. Pada tugas akhir kali ini akan digunakannya metode *load balancing* yang disebut dengan *Round Robin. Round Robin Algorithm* adalah algoritma paling sederhana dan paling banyak digunakan. Di dalam algoritma *Round Robin*, proses membagi waktu CPU dengan mengalokasikan waktu untuk setiap proses yang disebut waktu kuantum (Qt). Panjang waktu kuantum biasanya antara 10 sampai 100 milidetik. Jika kuantum yang sedang berlangsung sedang tidak kosong, eksekusi akan dihentikan sementara dan antrian berikutnya ditempatkan pada antrian yang tidak sibuk atau dapat dikatakan siap menerima *request*. Prosedur ini diulang hingga menyelesaikan eksekusi tiap antrian. Selain itu, proses yang telah selesai akan dihapus dari antrian. [6].



Round Robin Algorithm (D)

III. METODE

Sistematika yang digunakan untuk penulisan tugas akhir adalah sebagai berikut:

A. BAB I PENDAHULUAN

Berisikan pembahasan latar belakang, rumusan masalah, tujuan, batasan masalah, metode penelitian, skema penulisan, dan jadwal pelaksanaan.

B. BAB II DASAR TEORI

Pada bab ini menjelaskan mengenai teknologi yang akan digunakan mulai dari *stack MERN* akan dijelaskan tiap komponennya, seperti apa itu *MySQL, ExpressJs, ReactJs, NodeJs*, kemudian penjelasan sistem *load balance* yang digunakan.

C. BAB III MODEL PERANCANGAN SISTEM

Pada bab ini menjelaskan tentang perancangan *website* seperti *flowchart* pembuatan *website*; bentuk model database yang akan digunakan menggunakan UML; dan bentuk model *load balance* yang akan digunakan.

D. BAB IV HASIL DAN ANALISIS

Pada bab ini berisi hasil dan analisis keseluruhan dari pengujian yang telah dilakukan mulai dari berhasil tidaknya *website* yang dibuat, berhasil tidaknya sistem *load balance* yang diterapkan pada *website*, dan melakukan survey untuk melakukan penilaian kemudahan akses *website* kepada pengguna acak tak terbatas usia.

E. BAB V KESIMPULAN

Pada bab ini berisi kesimpulan dari seluruh analisis dan hasil penelitian yang dilakukan, serta pada bab ini dilengkapi dengan saran untuk penelitian selanjutnya.

Dalam melakukan penelitian yang telah dilakukan perancangan sebelumnya, diperlukannya target pelaksanaan penelitian. Berikut rancangan target pengerjaan penelitian:

Tabel 1. 1 Tabel Jadwal Pelaksanaan

No.	Deskripsi Tahapan	Durasi	Tanggal Selesai	*Milestone*
1	Desain sistem	2 minggu	Bulan ke-1 minggu ke-2	Riset dan pembuatan diagram blok
2	Perancangan perangkat lunak (*website*)	3 minggu	Bulan ke-2 minggu ke-1	Perancangan database menggunakan Firestore Database, desain *load balance* menggunakan metode *Round Robin*
3	Pembuatan perangkat lunak (*website)*	1 bulan	Bulan ke-3 minggu ke-1	Pembuatan *website* dengan keamanan, *database*, dan fitur *load balance* yang sudah dirancang
4	Pengujian dan analisis	1 bulan	Bulan ke-4 Minggu ke-1	Pengujian fitur *website*, pengambilan data, dan pembuatan kesimpulan

IV. HASIL DAN PEMBAHASAN

A. Hasil Pengujian Performansi Beban Server

Dari pemaparan *data* pada **Tabel 1** dan **Gambar 1** didapatkan tingkat persentase *error request* pada pengujian "**API Prediksi**". Secara berurutan dengan *connections* 332, 625, dan 800 menghasilkan *error request* sebagai berikut:

1. 0%, 20.488%, 32.954% untuk *server* tanpa *load balancer*,
2. 0%, 0%, 0.3% Suntuk *server* dengan *load balancer*

Parameter yang digunakan untuk melihat perbandingan antara server yang menggunakan load balancer dengan yang tidak adalah error request. Dari pengujian yang dilakukan, lima kali percobaan stress test pada API menghasilkan hasil rata-rata bahwa server yang menggunakan load balancer memiliki kamampuan handle request lebih besar, hal tersebut dapat ditarik kesimpulan oleh penulis karena pada server yang menggunakan load balancer hanya mengalami error request ketika jumlah requestnya 800. Faktor yang dapat menghasilkan error request pada tiap server adalah koneksi internet dan kapasitas layan dari server. Dari pernyataan tersebut, penulis menyimpulkan bahwa load balancer memiliki pengaruh pada server.

B. Hasil Pengujian *Throughput*

Dari pemaparan *data* pada **Tabel 2** dan **Gambar 2, Gambar 3,** dan **Gambar 4** didapatkan *throughput (request per second, received, and sent Kilo Bytes per second)* untuk API prediksi berurutan dengan *connections* 332, 625, dan 800 yaitu:

1. 44.754 *KB/sec*, 172.474 *KB/sec*, 268.538 *KB/sec (received)*; 61.686 *KB/sec*, 71.156 *KB/sec*, 66.396 *KB/sec (sent)*; dan 199.2588 *req/sec*, 287.0856 *req/sec*, 316.7676 *req/sec* (*request per second*) untuk *server* tanpa *load balancer*,
2. 92.44 *KB/sec*, 110.124 *KB/sec*, 82.426 *KB/sec (received)*; 127.404 *KB/sec*, 151.782 *KB/sec*, 110.418 *KB/sec (sent)*; dan 411.5525 *req/sec*, 490.2939 *req/sec*, 357.8175 *req/sec* (*request per second*) untuk *server* dengan *load balancer*.

Setelah dilakukan percobaan, penulis mendapatkan hasil berupa data yang dapat dilihat pada **Tabel 2**. Dari tabel tersebut penulis membuat grafik untuk membandingkan nilai throughput (received). throughput (sent), dan throughput (request per second) yang berturut-turut dapat dilihat pada **Gambar 2**, **Gambar 3**, dan **Gambar 4**.

Pada **Gambar 2** dapat dilihat bahwa perbedaan antara server yang menggunakan load balancer dan yang tidak menggunakan load balancer memiliki sifat yang berbeda. Server yang menggunakan load balancer nilainya lebih terlihat stabil dan rendah, dibandingkan dengan yang tidak menggunakan load balancer, memiliki nilai selalu naik dan tidak stabil. Faktor yang mempengaruhi hal tersebut kemungkinan adalah jenis request yang diuji adalah POST method sehingga untuk throughput (received) kali ini tidak terlalu besar nilainya. Pada grafik ini penulis dapat menyimpulkan bahwa server yang menggunakan load balancer akan menghasilkan throughput lebih stabil.

Pada **Gambar 3** dapat dilihat bahwa perbedaan antara server yang menggunakan load balancer dan yang tidak menggunakan load balancer memiliki perbedaan yang signifikan, nilai throughput (sent) yang dihasilkan oleh server yang menggunakan load balancer nilainya hampir dua kali lipat dari yang tidak menggunakan load balancer. Hal tersebut sudah disinggung pada paragraf sebelumnya, yaitu jenis request yang digunakan adalah POST method, sehingga throughput (sent) akan lebih dibutuhkan disini.

Pada **Gambar 4** dapat dilihat bahwa perbedaan antara server yang menggunakan load balancer dan yang tidak menggunakan load balancer memiliki perbedaan yang signifikan. Throughput (request per second) untuk server yang menggunakan load balancer memiliki nilai yang besar, hal tersebut dapat disimpulkan penulis, bahwa server yang menggunakan load balancer dapat menangani permintaan dari pengguna lebih banyak.

Dari tiga grafik di atas dapat disimpulkan bahwa penggunaan load balancer pada server sangat berpengaruh.

C. Hasil Pengujian *Delay*

Dari pemaparan *data* pada **Tabel 2** dan **Gambar 5** didapatkan *delay* berurutan pada API prediksi dengan *connections* 332, 625, dan 800 yaitu:

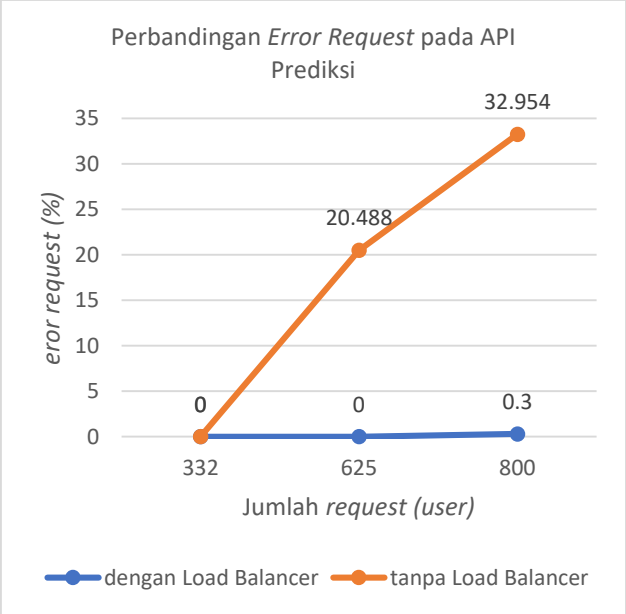
1. 690.2 ms, 734.4 ms, 682.4 ms untuk *server* tanpa *load balancer*,
2. 82.8 ms, 155.6 ms, 103 ms untuk *server* dengan *load balancer*.

Setelah dilakukan percobaan, penulis mendapatkan hasil berupa data yang dapat dilihat pada **Tabel 2**. Dari tabel tersebut penulis membuat grafik untuk membandingkan nilai delay dapat dilihat pada **Gambar 5**.

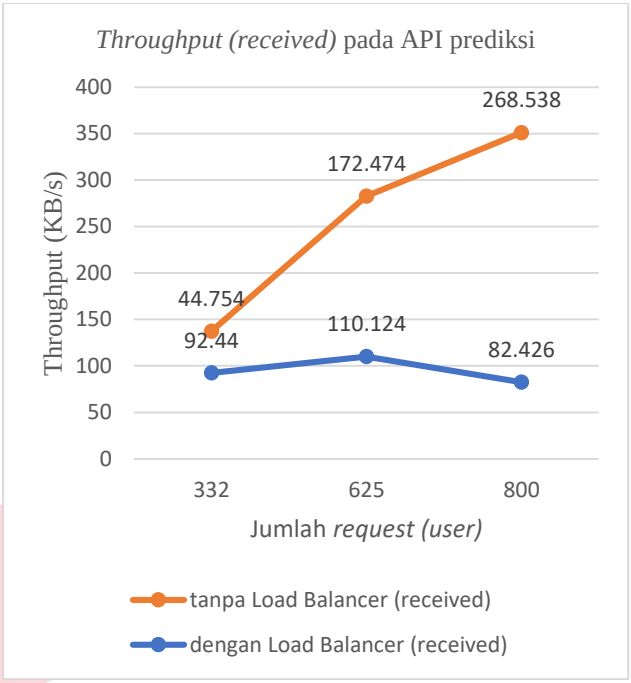
Dari grafik pada **Gambar 5** dapat dilihat nilai delay pada tingkat penanganan server yang menggunakan load balancer lebih tinggi dibandingkan dengan yang tidak menggunakan load balancer. Hal tersebut dikarenakan bahwa server yang tidak menggunakan load balancer akan mengalami kesulitan dalam melayani jumlah permintaan dari pengguna yang berjumlah besar dan akan menimbulkan antrean yang panjang yang akan menyebabkan nilai delay makin besar.

Dari dua pernyataan di atas dapat disimpulkan bahwa penggunaan load balancer pada server akan membantu meringankan beban server yang awalnya hanya dikerjakan satu server menjadi dikerjakan oleh tiga server.

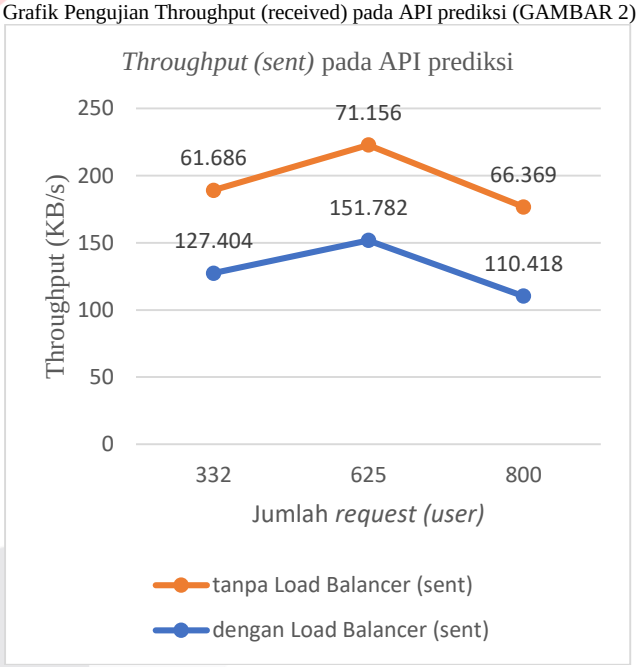
1. Gambar



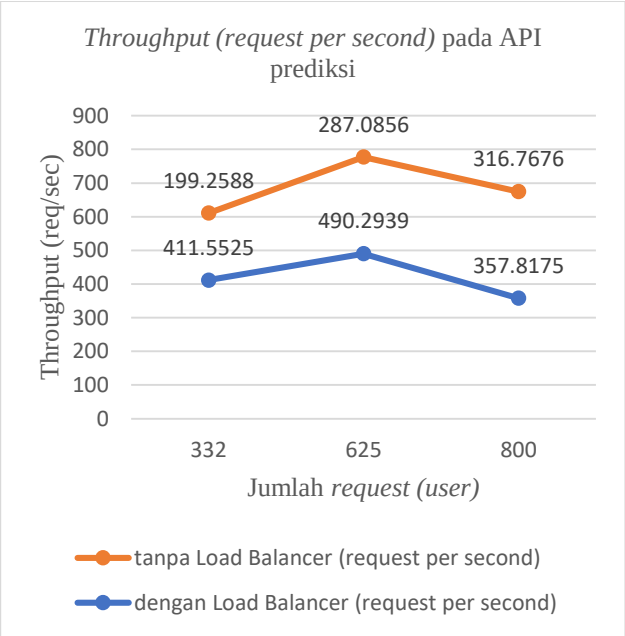
Perbandingan Error Request pada API Prediksi (GAMBAR 1)



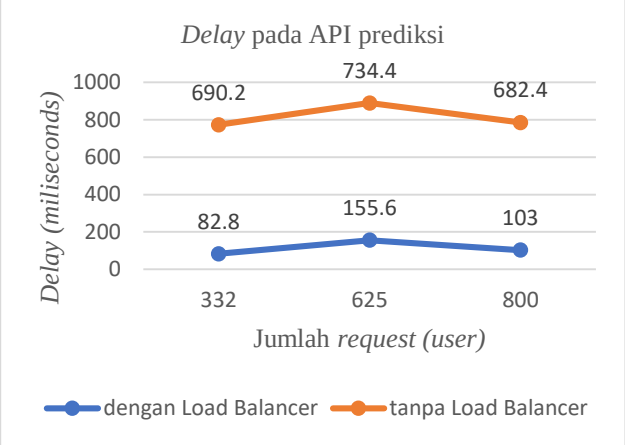
Grafik Pengujian Throughput (received) pada API prediksi (GAMBAR 2)



Grafik Pengujian Throughput (sent) pada API prediksi (GAMBAR 3)



Grafik Pengujian Throughput (request per second) pada API prediksi (GAMBAR 4)



Grafik Pengujian Delay pada API prediksi (GAMBAR 5)

2. Tabel

Hasil pengujian performansi beban server pada API prediksi (TABEL 1)

	Thread	*Error Request (%)*
Tanpa *Load Balancer*	332	0
	625	20.488
	800	32.954
Dengan *Load Balancer*	332	0
	625	0
	800	0.3

Hasil pengujian Quality of Service pada API prediksi (TABEL 2)

	Thread	*Average Latency (ms)*	*Throughput Req/Sec*	*Throughput (received) Kb/Sec*	*Throughput (sent) Kb/Sec*
Tanpa *Load Balancer*	332	690.2	199.2588	44.754	61.686
	625	734.4	287.0856	172.474	71.156
	800	682.4	316.7676	268.538	66.396
Dengan *Load Balancer*	332	82.8	411.5525	92.44	127.404
	625	155.6	490.2939	110.124	151.782
	800	103	357.8175	82.426	110.418

V. KESIMPULAN

Kemudian penulis melakukan pengujian dari rancangan dan implementasi yang telah dibuat. Dari pengujian tersebut penulis dapat menyimpulkan bahwa:

A. Pada *website* ini menyediakan prediksi untuk waktu kedatangan bus dari halte berangkat menuju halte tujuan yang dapat ditentukan oleh penumpang ataupun calon penumpang TMB. Selain itu, *website* ini juga menyediakan prediksi kemacetan dari jalur yang akan dilewati bus armada TMB dari halte ke halte.

B. *Website* yang menggunakan *load balancer* dapat meminimalisir tingkat *error request*, hal tersebut dapat disimpulkan berdasarkan pengujian *server* yang tidak menggunakan *load balancer* dan yang menggunakan *load balancer*. Percobaan tersebut dilakukan dengan melakukan percobaan *stress test* dengan jumlah *request* 332, 625, dan 800 berturut-turut menghasilkan yang awalnya memiliki nilai sebesar 0%, 26.19%, 34.63% menjadi 0%, 0%, o%, sehingga, dinilai bahwa *load balancer* berdampak pada kemampuan pelayanan *server*.

C. Pengujian *quality of service* pada bagian *throughput (received)* mendapatkan hasil bahwa *server* yang menggunakan *load balancer* memiliki nilai yang stabil tetapi rendah yaitu 92.44 *KB/sec*, 110.124 *KB/sec*, 82.426 *KB/sec* sedangkan untuk *server* yang tidak menggunakan *load balancer* menghasilkan nilai *throughput (received)* 44.754 *KB/sec*, 172.474 *KB/sec*, 268.538 *KB/sec*. Kemudian untuk pengujian *throughput (sent) server* yang menggunakan *load balancer* terlihat lebih tinggi dikarenakan jenis *request* yang diuji adalah *POST method* yaitu sebesar 127.404 *KB/sec*, 151.782 *KB/sec*, 110.418 *KB/sec* dibandingkan dengan yang tidak menggunakan *load balancer* sebesar 61.686 *KB/sec*, 71.156 *KB/sec*, 66.396 *KB/sec*. Lalu untuk pengujian *throughput (request per second)* menghasilkan *data* untuk *server* yang menggunakan *load balancer* memiliki kapasitas tampung layanan lebih besar yaitu sebesar 411.5525 *req/sec*, 490.2939 *req/sec*, 357.8175 *req/sec* dibandingkan dengan yang tidak menggunakan *load balancer* sebesar 199.2588 *req/sec*, 287.0856 *req/sec*, 316.7676 *req/sec*. Dari ketiga jenis pengujian *throughput* tersebut dapat disimpulkan *server* yang menggunakan *load balancer* dapat menghasilkan *throughput (sent)* lebih besar dan daya tampung yang lebih banyak.

D. Pengujian *quality of service* pada bagian *delay menghasilkan* bahwa *server* yang menggunakan *load balancer* dapat memecah antrean dan memperkecil nilai *delay* yang awalnya sebesar 690.2 ms, 734.4 ms, 682.4 ms menjadi 82.8 ms, 155.6 ms, 103 ms.

REFERENSI

Electronic References

[1] K. C. Amalia, R. Munadi and N. Karna, "PERANCANGAN DAN IMPLEMENTASI WEB UNTUK PRODUK S-LUCY (SMART LIGHT ULTIMATE CONTROL BY WEBSITE) BERBASIS INTERNET OF THINGS," 2021.

[2] A. Javeed, "Performance Optimization Techniques for ReactJS," *2019 IEEE International Conference*

on Electrical, Computer and Communication Technologies (ICECCT), pp. 1-5, 2019.

[3] W.-J. Li, C. Yen, Y.-S. Lin, S.-C. Tung and S. Huang, "JustIoT Internet of Things based on the Firebase Real-time Database," p. 44, 2018.

[4] S. Shukla, C. S. Gupta and P. Mishra, "Android-Based Chat Application Using Firebase," *International Conference Computer Communication and Informatics*, 2021.

[5] A. M. Komaruddin1, D. M. Sipitorini and P. Rispian, "LOAD BALANCING DENGAN METODE ROUND ROBIN UNTUK PEMBAGIAN BEBAN KERJA WEB SERVER," *Jurnal Siliwangi*, vol. V, no. 2, pp. 47-50, 2019.

[6] T. Balharith and F. Alhaidari, "Round Robin Scheduling Algorithm in CPU and Cloud Computing : A review," *2019 2nd International Conference on Computer Applications & Information Security (ICCAIS)*, vol. 2, pp. 1-7, 2019.

[7] H. Hassani, M. Ghodsi and G. Howell, "A Note on Standard Deviation and Standard Error," *Teaching Mathematics and Its Application*, p. 112, 2010.

[8] Listifadah and R. Puspitasari, "EVALUASI KINERJA TRANS METRO BANDUNG," *Penelitian Transportasi Darat*, vol. 17, no. 2, pp. 65-78, 2015.