Jurnal Pendidikan dan Teknologi Indonesia (JPTI)
Vol. 4, No. 11, November 2024, Hal. 345-358
DOI: https://doi.org/10.52436/1.jpti.477
p-ISSN: 2775-4227
e-ISSN: 2775-4219

Implementasi Sistem *Load Balancing* Pada Web Server Berbasis Raspberry Pi Dengan Metode *Long Short-Term Memory*

Halimatusyadiah[*1], Yuggo Afrianto[2], Bayu Adhi Prakosa[3]

[1,2,3]Informatics Department, Faculty of Engineering and Science, Universitas Ibn Khaldun Bogor, Indonesia
Email: [1]syadiah128@gmail.com, [2]yuggo@ft-uika-bogor.ac.id, [3]bayu.adhi@uika-bogor.ac.id

Abstrak

Peningkatan jumlah pengguna internet telah berhasil meningkatkan beban pada web server, mengakibatkan risiko *overload* dan penurunan kinerja layanan. Sistem *load balancing* menjadi solusi penting untuk mendistribusikan permintaan layanan secara merata. Penelitian ini mengembangkan sistem *load balancing* berbasis metode *Long Short-Term Memory* (LSTM) pada server web dengan menggunakan Raspberry Pi sebagai platform komputasi. Metode ini dijalankan untuk memprediksi beban CPU berdasarkan data *time series*, memungkinkan distribusi beban yang lebih akurat dan dioptimalkan. Hasil pengujian menunjukkan peningkatan total permintaan sebesar 4%, penurunan waktu respons rata-rata sebesar 9%, pengurangan tingkat error sebesar 32%, dan peningkatan *throughput* sebesar 12%. Penelitian ini memberikan kontribusi penting pada pengembangan sistem *load balancing* berbasis prediksi yang lebih disesuaikan terhadap pola lalu lintas dinamis, sekaligus menawarkan solusi hemat biaya untuk implementasi pada skala kecil.

Kata kunci: *Load Balancing, Long Short-Term Memory, Raspberry Pi, Time Series, Web Server*

Implementation of Load Balancing System on Raspberry Pi Based Web Server with Long Short-Term Memory Method

Abstract

The increase in the number of internet users has successfully increased the load on web servers, resulting in the risk of overload and decreased service performance. Load balancing system becomes an important solution to distribute service requests evenly. This research develops a load balancing system based on the Long Short-Term Memory (LSTM) method on a web server using Raspberry Pi as a computing platform. This method is run to predict CPU load based on time series data, enabling more accurate and optimized load distribution. The test results show an increase in total requests by 4%, a decrease in average response time by 9%, a reduction in error rate by 32%, and an increase in throughput by 12%. This research makes an important contribution to the development of prediction-based load balancing systems that are more adapted to dynamic traffic patterns, while offering a cost-effective solution for small-scale implementation.

Keywords: *Load Balancing, Long Short-Term Memory, Raspberry Pi, Time Series, Web Server*

1. PENDAHULUAN

Pentingnya internet dalam kehidupan modern tidak dapat untuk diabaikan. Dengan semakin banyak orang mengandalkan internet untuk mencari informasi, melakukan transaksi, dan promosi, kebutuhan akan web server yang andal menjadi sangat penting [1]. Web server digunakan untuk memenuhi permintaan data dari pengguna melalui protokol HTTP atau HTTPS, memungkinkan akses terhadap berbagai media digital. Namun, seiring bertambahnya jumlah pengguna, beban pada web server meningkat signifikan, yang sering kali menyebabkan *overload* dan penurunan kinerja [2].

Salah satu cara utama untuk mengatasi masalah ini adalah melalui sistem *load balancing*. Dengan mengalokasikan permintaan layanan secara merata di antara beberapa server, *load balancing* meningkatkan ketersediaan dan kinerja layanan [3]. Dengan mengurangi beban pada setiap server, metode ini mengurangi kemungkinan kegagalan server yang disebabkan oleh kelebihan beban [4].

Penelitian sebelumnya menunjukkan bahwa algoritma seperti Round Robin dan Least Connection sering digunakan dalam sistem *load balancing*. Algoritma tersebut efektif untuk distribusi beban statis tetapi kurang adaptif terhadap pola lalu lintas yang dinamis, terutama dalam skenario dengan variasi beban yang cepat berubah

p-ISSN: 2775-4227
e-ISSN: 2775-4219

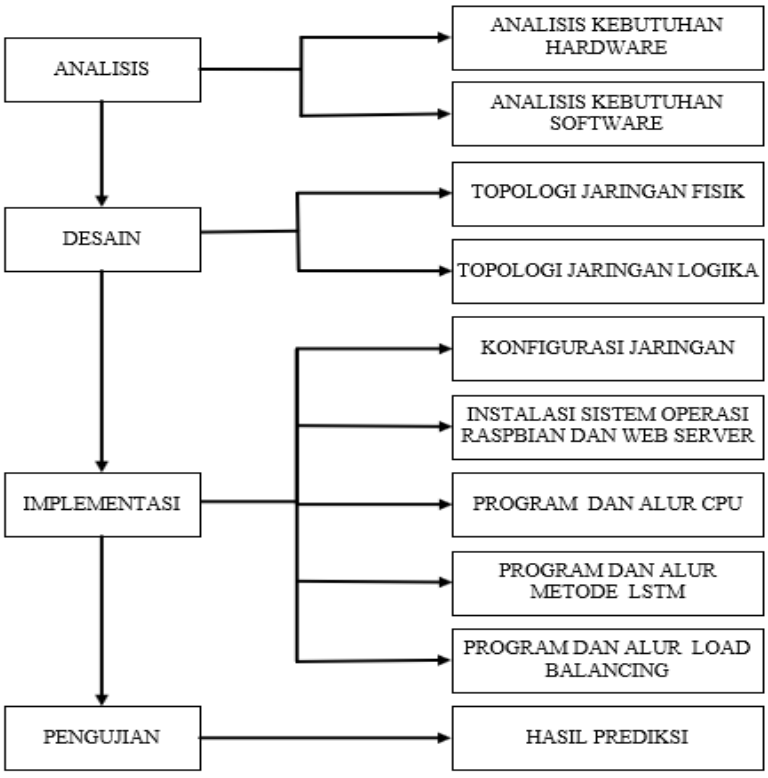
[5], [6]. Sebagai alternatif, metode *Long Short-Term Memory* (LSTM) diperkenalkan. LSTM merupakan jaringan saraf tiruan yang dirancang untuk memproses data *time series*. Metode ini memiliki keunggulan dalam mempertahankan informasi jangka panjang dan memberikan prediksi akurat pada data dengan pola kompleks [7][8].

Selain itu, Raspberry Pi menjadi perangkat pilihan untuk penelitian skala kecil karena sifatnya yang hemat energi, murah, dan fleksibel. Raspberry Pi 3 Model B, dengan prosesor *quad-core* dan dukungan sistem operasi Linux, mampu menjalankan server web sederhana sekaligus memproses algoritma prediktif seperti LSTM [9]. Raspberry Pi dikenal sebagai server utama yang hemat biaya, hemat energi, dan fleksibel, sehingga ideal untuk penelitian skala kecil. Kombinasi Raspberry Pi dan metode LSTM menawarkan pendekatan baru dalam mengoptimalkan distribusi beban server web. LSTM memungkinkan prediksi beban server yang akurat, sementara Raspberry Pi memberikan platform komputasi yang efisien untuk penelitian skala laboratorium [10].

Oleh karena itu, penelitian ini bertujuan untuk mengembangkan sistem *load balancing* berbasis metode LSTM pada server web menggunakan Raspberry Pi. Sistem ini dirancang untuk meningkatkan efisiensi distribusi beban dan mengurangi kemungkinan *server overload*, sekaligus menawarkan solusi hemat biaya yang adaptif terhadap pola lalu lintas dinamis.

2. METODE PENELITIAN

Isi Metode penelitian ini merupakan tahapan yang disusun sistematis bertujuan dalam memperoleh hasil yang baik. Berikut adalah langkah-langkah yang harus dilakukan sebelum mengimplementasikan sistem *load balancing* web server menggunakan metode *long short-term memory* pada Gambar 1.



Gambar 1. Metode Penelitian

Pada tahap-tahap yang terdapat pada gambar 1 akan dijelaskan sebagai berikut:

a. Analisis

Fase pertama dimulai dengan fase analisis atau Analisa yaitu proses menganalisa dan menetapkan apa saja yang dibutuhkan, dalam proses implementasi sistem *load balancing* web server menggunakan metode *Long Short-Term Memory* (LSTM). Untuk membuat sistem *load balancing* server web dengan menggunakan pendekatan *Long Short-Term Memory* (LSTM), langkah adalah mempelajari proses dan menentukan yang dibutuhkan [11]. Tahap ini dilakukan untuk mengidentifikasi kebutuhan perangkat keras dan perangkat lunak. Perangkat keras meliputi Raspberry Pi, Router Mikrotik, dan laptop, sedangkan perangkat lunak mencakup sistem operasi Raspbian, Python, dan perangkat konfigurasi jaringan seperti Winbox.

p-ISSN: 2775-4227
e-ISSN: 2775-4219

b. Desain

Pada tahap ini, dirancang topologi fisik dan logis jaringan. Topologi fisik menggambarkan koneksi perangkat melalui Ethernet dan Wi-Fi, sedangkan topologi logis mencakup pengaturan alamat IP dan subnet.

c. Implementasi

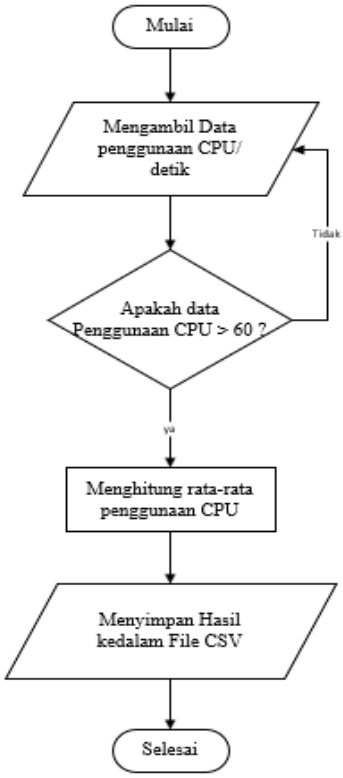
Sistem operasi Raspbian diinstal pada Raspberry Pi untuk menjalankan server web. Algoritma LSTM diprogram menggunakan Python untuk memprediksi penggunaan CPU berdasarkan data time series. Data dikumpulkan setiap detik selama 60 detik dan disimpan dalam file CSV.

d. Pengujian

Pengujian dilakukan dengan simulasi lalu lintas menggunakan Apache JMeter, dengan skenario berbeda untuk mengukur respons server. Hasil diukur menggunakan metrik seperti RMSE, MAE, dan MAPE untuk mengevaluasi akurasi model prediksi.

2.1. CPU

Proses dimulai dengan mengumpulkan data persentase penggunaan CPU setiap detik hingga terkumpul 60 data atau selama 60 detik. Setelah itu, rata-rata penggunaan CPU dihitung, dan hasil perhitungan tersebut beserta waktu penyelesaian perhitungan disimpan dalam file berformat CSV. Program monitoring penggunaan CPU ini dirancang untuk memantau dan merekam data penggunaan CPU pada komputer atau server. Penelitian ini *persentase* penggunaan CPU direkam setiap detik. Setelah 60 data terkumpul (setara dengan 60 detik), rata-rata penggunaan CPU dihitung dan direkam berdasarkan waktu penyelesaian perhitungan untuk menghasilkan data urut waktu *(time series)*. Data yang terkumpul kemudian disimpan dalam file CSV yang terdiri dari dua kolom: waktu dan rata-rata penggunaan CPU. Program monitoring CPU berbasis Python ini dijalankan pada setiap komputer server.

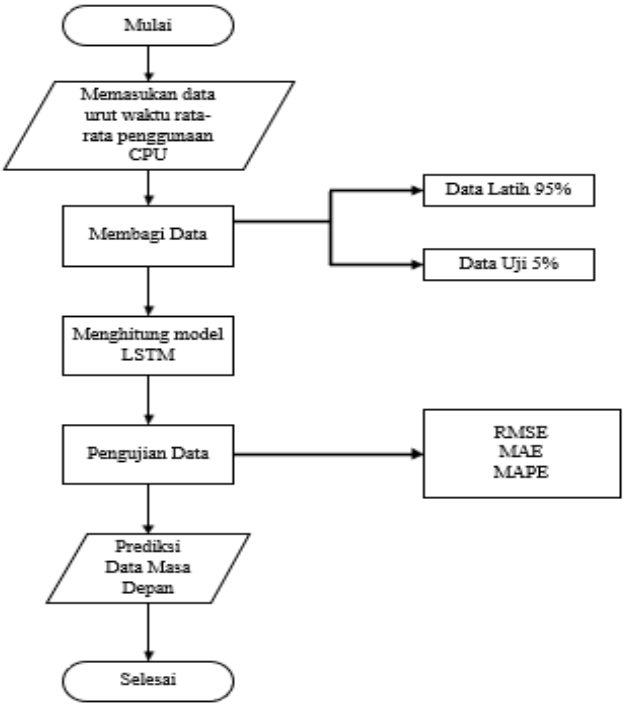


Gambar 2. Tahap Alur CPU

2.2. LSTM

Program perhitungan LSTM yang dibuat ini. dimulai dengan memasukan data rata-rata penggunaan CPU yang dipantau, lalu dibagi menjadi dua data yaitu data latih dan data uji, selanjutnya mencari dan menghitung model LSTM. Model yang sudah didapatkan, dihitung matrik kesalahannya menggunakan RSME *(Root Mean Squared Error)*, MSE *(Mean Squared Error)*, dan MAE *(Mean Absolute Error)*. Jika sudah tahap selanjutnya adalah memprediksi jumlah data masa depan atau yang akan datang.

p-ISSN: 2775-4227
e-ISSN: 2775-4219



Gambar 3. Tahap Alur LSTM

2.3. *Time Series*

Deret waktu atau *time series* adalah catatan pengamatan satu atau beberapa variabel yang diambil secara berurutan dalam interval waktu yang tetap. Pada tahun 1976, George Box dan Gwilyn Jenkins memperkenalkan analisis *time series* untuk pertama kalinya. Prosedur statistika ini digunakan untuk meramalkan peristiwa di masa depan dengan memanfaatkan data yang terorganisir menurut urutan waktu. Analisis *time series* memperhitungkan korelasi antara peristiwa saat ini dengan periode waktu sebelumnya. Selain hubungan waktu, *time series* juga dapat menunjukkan korelasi dengan dimensi lain, seperti wilayah atau dimensi lainnya yang saling terkait [12].

2.4. *Error Metrics*

Error metrics digunakan untuk mengevaluasi performa model peramalan atau prediksi dalam memprediksi nilai sebenarnya. Ukuran kesalahan ini menilai sejauh mana perkiraan model mendekati nilai aktual dan seberapa akurat model tersebut dalam memprediksi nilai yang sebenarnya. Metode yang digunakan untuk mengukur kesalahan termasuk RMSE, MAE dan MAPE [13].

2.5. RMSE

Root Mean Square Error (RMSE) adalah akar kuadrat dari *Mean Square Error* (MSE) dan merupakan metrik yang sering digunakan untuk mengukur rata-rata perbedaan kuadrat antara nilai prediksi dan nilai aktual. RMSE sering dianggap sebagai ukuran yang lebih representatif untuk kesalahan "standar" karena memiliki satuan yang sama dengan variabel yang diukur, membuatnya lebih intuitif untuk interpretasi dalam konteks yang sama dengan data aslinya [14].

$$\frac{\sum_{t=1}^{n}\left[\frac{a-b}{a}\right]}{n} \times 100\% \qquad (1)$$

2.6. MAE

Mean Absolute Error (MAE) adalah metrik yang sederhana namun efektif untuk mengukur kesalahan prediksi model, terutama dalam kasus di mana terdapat outlier atau distribusi kesalahan yang tidak normal.

p-ISSN: 2775-4227
e-ISSN: 2775-4219

Metrik ini sering digunakan bersamaan dengan RMSE untuk memberikan gambaran yang lebih komprehensif tentang kinerja model [14].

$$\frac{1}{n}\sum_{t=1}^{n}|y_t - y_t| \quad (2)$$

2.7. MAPE

Mean Absolute Percentage Error (MAPE) adalah ukuran yang digunakan untuk menilai akurasi model peramalan. MAPE dihitung dengan mengambil nilai absolut dari kesalahan pada setiap periode, membaginya dengan nilai observasi aktual untuk periode tersebut, kemudian merata-ratakan hasilnya. Nilai ini dinyatakan dalam *persentase*, sehingga memudahkan untuk menginterpretasikan besarnya kesalahan relatif terhadap nilai sebenarnya [15].

$$\frac{\sum_{t=1}^{n}\left[\frac{a-b}{a}\right]}{n} \times 100\% \quad (3)$$

3. HASIL DAN PEMBAHASAN

Hasil penelitian ini melibatkan pengujian sistem *load balancing* berbasis metode *Long Short-Term Memory* (LSTM) yang diimplementasikan pada Raspberry Pi. Hasil pengujian mencakup analisis peningkatan performa server, yang dijelaskan melalui tabel, grafik, dan diskusi berikut.

3.1. Analisis

Untuk menyelesaikan langkah analisis, ada dua persyaratan yang harus dipenuhi, yaitu:

a. Kebutuhan *Hardware*

Untuk mengembangkan sistem *load balancing* server web dengan menggunakan *Long Short-Term Memory* (LSTM) membutuhkan perangkat keras, seperti yang terlihat pada Tabel 1.

Tabel 1. Kebutuhan *Hardware*

Perangkat Keras	Fungsi / Kegunaan
Laptop	Digunakan sebagai klien, untuk menjalankan aplikasi algoritma *load balancing server* web, dan untuk menghitung data deret waktu menggunakan pendekatan LSTM.
Router Mikrotik	Digunakan untuk menautkan dan mengontrol jaringan server web *Raspberry Pi.*
Raspberry Pi.	Digunakan sebagai *server* web dan untuk mengumpulkan data deret waktu rata-rata penggunaan CPU.

b. Kebutuhan *Software*

Studi yang dilakukan pada perangkat lunak untuk pendekatan *Long Short-Term Memory* (LSTM) dalam implementasi *load balancing* server web, seperti yang terlihat pada Tabel 2.

Tabel 2. Kebutuhan *Software*

Perangkat Lunak	Fungsi / Kegunaan
Raspbian	Sistem operasi yang digunakan oleh perangkat Raspberry Pi.
Visual Studio Code	Editor kode yang dibuat oleh Microsoft, sebuah bisnis perangkat lunak sumber terbuka. Editor ini digunakan untuk mengembangkan dan menjalankan aplikasi sistem *load balancing server* web berbasis LSTM.
Winbox	Program yang memungkinkan konfigurasi *router proxy* dengan tampilan *User Interface* (UI) yang lebih baik.

3.2. Desain

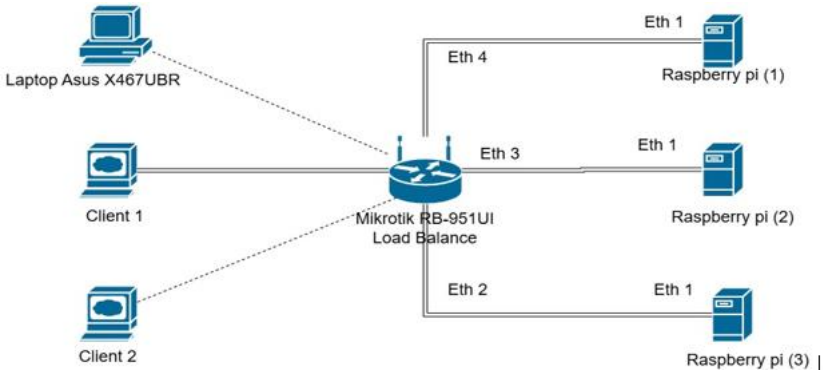
Pendekatan *Long Short-Term Memory* (LSTM) digunakan pada tahap desain, arsitektur jaringan, dan implementasi sistem load balancing web server.

1. Topologi Jaringan Fisik

Arsitektur struktur jaringan lokal yang dibangun oleh peneliti di ruang laboratorium Sistem dan Jaringan Komputer diilustrasikan dengan topologi jaringan fisik, seperti yang ditunjukkan pada gambar 4. Dengan koneksi kabel ke Raspberry Pi dan koneksi nirkabel ke laptop ASUS X407UBR yang berfungsi sebagai dan

p-ISSN: 2775-4227
e-ISSN: 2775-4219

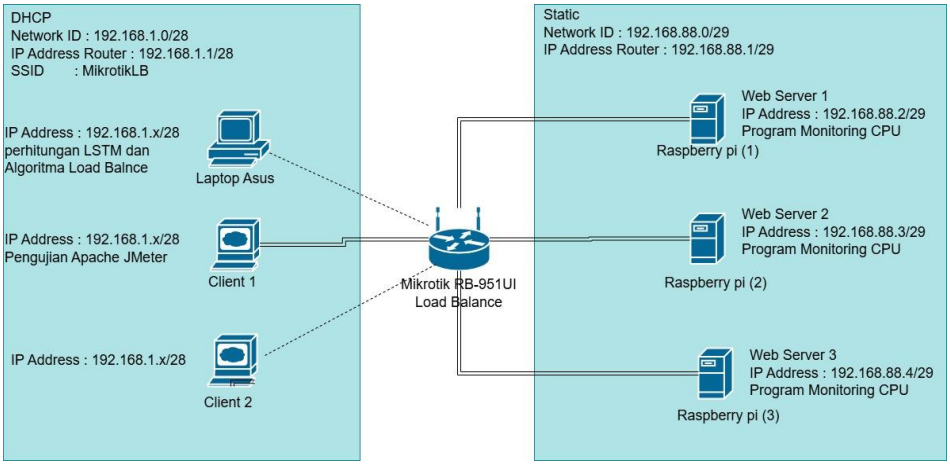
mikrokontroler untuk menjalankan perintah load balancing dan menghitung kinerja CPU dengan metode LSTM, jaringan lokal ini menggunakan *router board* Mikrotik RB-951UI sebagai pusat jaringan dan menjalankan *load balancing*. Pada konfigurasi ini, Raspberry Pi 1 terhubung ke eth5, Raspberry Pi 2 ke eth4, dan Raspberry Pi 3 ke eth3. Setiap Raspberry Pi dihubungkan ke router menggunakan koneksi ethernet terpisah. Dua klien PC selanjutnya dihubungkan ke router secara nirkabel.



Gambar 4. Topologi Jaringan Fisik

2. Topologi Jaringan Logika

Gambar 5 mengilustrasikan arsitektur logis dari struktur jaringan lokal yang dibangun oleh para peneliti CSN Lab, yang mencakup pengalamatan alamat IP. Network ID 192.168.1.0/24 dan pengaturan DHCP digunakan oleh pengguna atau *client* yang terhubung secara *wireless* dengan SSID MikrotikLB, sehingga IP Address akan terdistribusi secara otomatis pada range 192.168.1.1-192.168.1.254/24 berdasarkan IP yang didapat dari router.



Gambar 5. Topologi Jaringan Logika

Network ID 192.168.88.0/29, yang terhubung secara nirkabel dan memiliki alamat IP statis yang dikonfigurasi, digunakan untuk jaringan Raspberry Pi. Memanfaatkan subnet mask 255.255.255.248, rentang alamat IP yang dapat digunakan dibatasi hingga 192.168.88.1-1922.168.88.7. 192.168.88.2/29 untuk Raspberry Pi 1, 192.168.88.3/29 untuk Raspberry Pi 2, dan 192.168.88.4/29 untuk Raspberry Pi 3 merupakan alamat IP dari Raspberry Pi yang digunakan sebagai web server. Sebuah laptop ASUS dengan alamat IP 192.168.1.X/24 dan subnet mask 255.255.255.248 digunakan sebagai mikrokontroler. Untuk menjaga keseimbangan beban melalui router untuk mentransfer beban lalu lintas dari klien ke server web, mikrokontroler ini akan menghitung prediksi LSTM dan mengirimkan instruksi perubahan jalur keseimbangan beban ke router. Untuk mencegah server kelebihan beban, hal ini memastikan bahwa trafik ke server web tersebar secara merata berdasarkan prediksi LSTM.

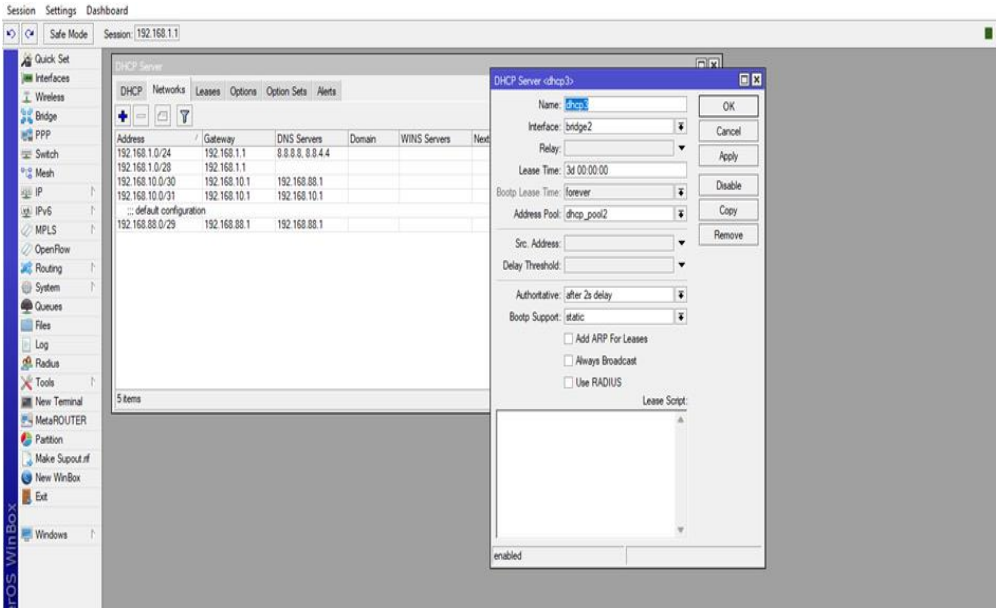
p-ISSN: 2775-4227
e-ISSN: 2775-4219

3.3. Implementasi

Pada Menerapkan konsep yang telah direncanakan ke dalam tindakan pada tahap ini. Peneliti sekarang memisahkan tahap implementasi menjadi beberapa fase.

a. Konfigurasi Jaringan

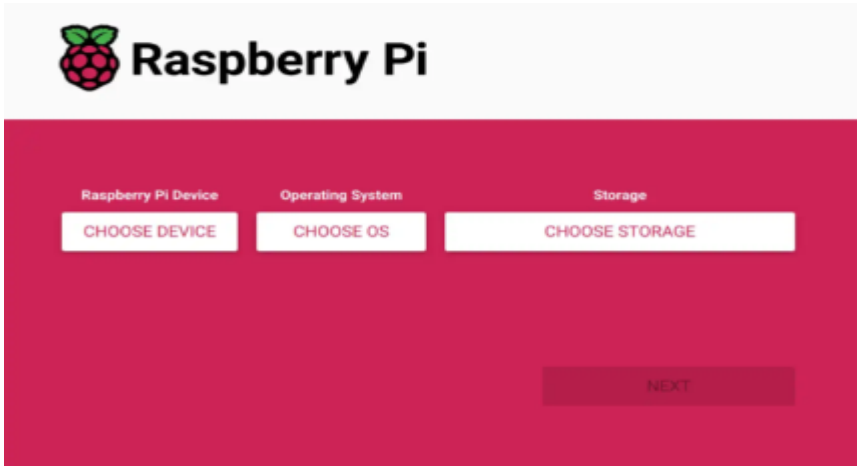
Pada Gambar 6 alamat IP didistribusikan ke setiap server atau klien untuk menyediakan akses internet. Ini menggambarkan bagaimana sebuah jaringan dikonfigurasikan. Subnet, gateway, dan server DNS harus dikonfigurasikan dengan hati-hati selama fase ini untuk menjamin komunikasi yang lancar antara perangkat. *Dynamic Host Configuration Protocol*, atau DHCP, juga dapat digunakan dalam konfigurasi yang ditampilkan untuk mengotomatiskan distribusi alamat IP, khususnya untuk perangkat klien [16].



Gambar 6. Konfigurasi Jaringan dan Mikrotik Via Winbox

b. Instalasi Sistem Raspbian dan Web Server

Pada gambar 7 tampilan pilih model sistem operasi Raspberry Pi yang ingin Anda gunakan, seperti Raspberry Pi OS (sebelumnya dikenal sebagai Raspbian) atau sistem operasi lain yang dapat diakses dalam daftar, setelah menjalankan aplikasi Raspberry Pi Imager. Pilih kartu microSD dari daftar perangkat yang ditemukan Raspberry Pi Imager, kemudian arahkan penyimpanan instalasi ke kartu yang telah terpasang ke komputer melalui pembaca kartu setelah memilih sistem operasi. Lanjutkan dengan prosedur flashing untuk menginstal sistem operasi yang dipilih ke dalam kartu microSD setelahnya [17].



Gambar 7. Tampilan Raspberry Pi

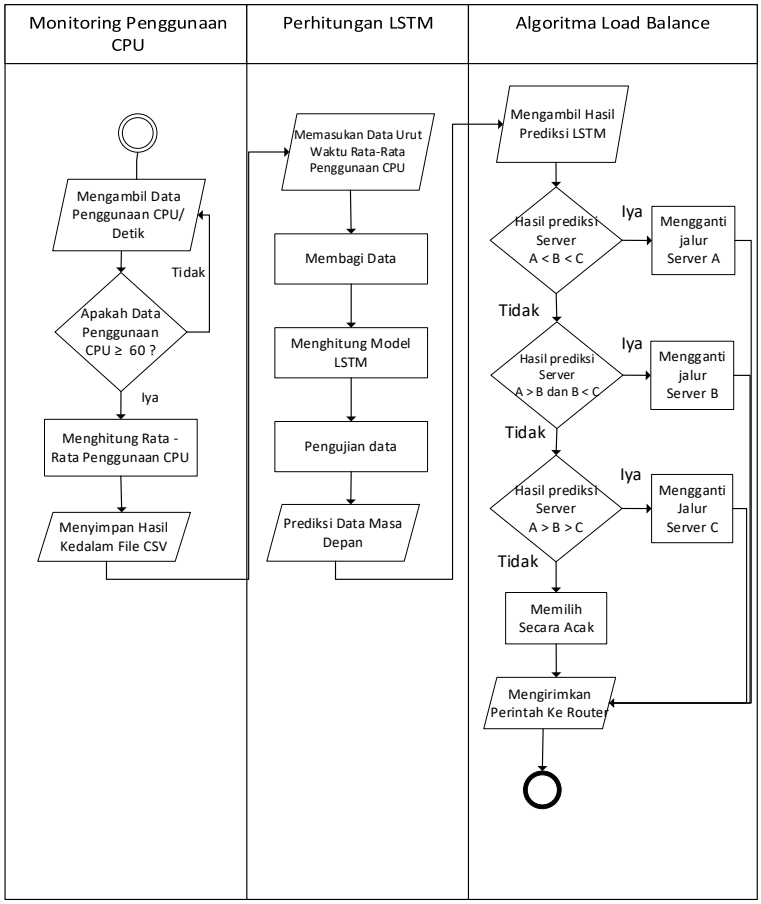
p-ISSN: 2775-4227
e-ISSN: 2775-4219

```
root@raspberrypi:/home/pi# apt install php php-common php-mysql php-snmp php-xml
 php-mbstring php-gd php-zip php-pear -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
php is already the newest version (2:7.4+76).
php-common is already the newest version (2:76).
php-gd is already the newest version (2:7.4+76).
php-mbstring is already the newest version (2:7.4+76).
php-mysql is already the newest version (2:7.4+76).
php-pear is already the newest version (1:1.10.12+submodules+notgz+20210212-1).
php-snmp is already the newest version (2:7.4+76).
php-xml is already the newest version (2:7.4+76).
php-zip is already the newest version (2:7.4+76).
The following package was automatically installed and is no longer required:
  libfuse2
Use 'sudo apt autoremove' to remove it.
0 upgraded, 0 newly installed, 0 to remove and 48 not upgraded.
root@raspberrypi:/home/pi# S
```

Gambar 8. Instalasi Web Server

Pada gambar 8 tampilan dalam hal manajemen basis data dalam pengembangan web dinamis, Apache dan PHP bekerja sama dengan baik. PHP berfungsi sebagai bahasa pemrograman sisi server yang memungkinkan interaksi basis data, sedangkan Apache berfungsi sebagai tulang punggung server web, mengelola permintaan HTTP dan mengirimkan konten *online* [18].

c. Program dan Alur CPU,LSTM, dan *Load Balancing* dilihat pada gambar 9.



Gambar 9. Tahap Alur Sistem

Setiap detik, perangkat lunak melacak penggunaan CPU, mengumpulkan data selama enam puluh detik, dan menghitung penggunaan CPU rata-rata, yang kemudian disimpan dalam file CSV. Model *Long Short-Term Memory* (LSTM) dilatih dan diuji menggunakan data ini untuk meramalkan beban server di masa depan. Metrik *Root Mean Squared Error* (RMSE), *Mean Squared Error* (MSE), dan *Mean Absolute Error* (MAE) digunakan

p-ISSN: 2775-4227
e-ISSN: 2775-4219

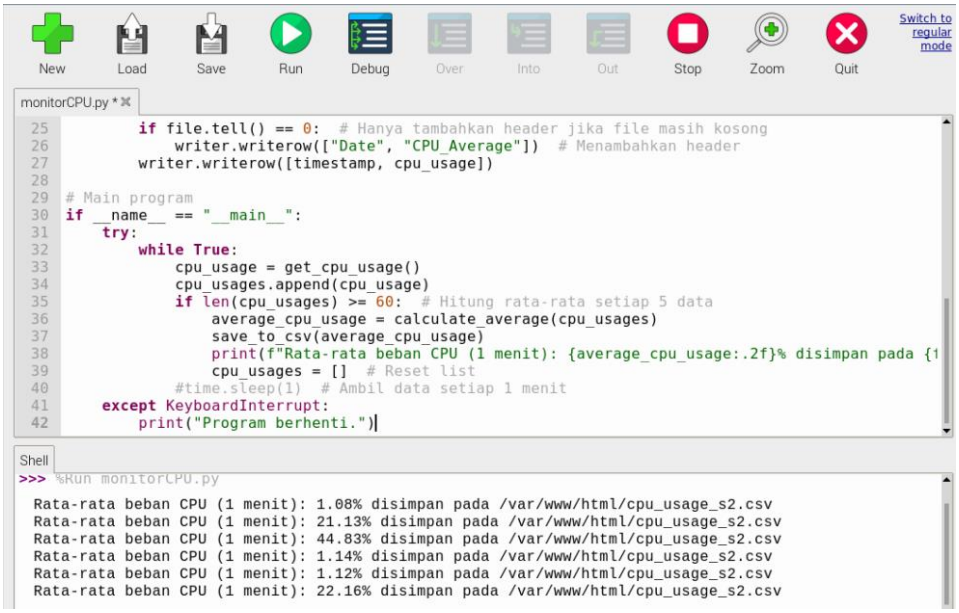
untuk mengukur kesalahan prediksi. Perkiraan beban server A, B, dan C dievaluasi selama proses penyeimbangan beban untuk menentukan server mana yang akan menerima trafik berdasarkan perkiraan beban terendah. Pemilihan acak digunakan jika setiap server memiliki perkiraan beban yang sama. Router menerima instruksi pengalihan trafik melalui SSH, dan memodifikasi pengaturan NAT untuk menemukan jalur ke server yang dipilih.

3.4. Pengujian

Tahapan ini merupakan pengujian dan hasil dari sistem *load balancing* pada *web server* berbasis raspberry pi dengan metode *long short-term memory*. Secara spesifik, tahap ini memberikan uraian mendetail mengenai efektifitas LSTM untuk penugasan server pada *load balancing*.

1. Pengujian dan Hasil Penggunaan CPU

Pada tahap ini, program pemantauan CPU yang telah dibuat diuji dengan menjalankannya di setiap *web server*, serta memeriksa *output* berupa file berformat CSV yang telah disimpan dapat dilihat pada gambar 10.



Gambar 10. Pengujian CPU

Pada gambar 10 program pengujian CPU disimpan dan dijalankan menggunakan kode editor Thonny Python di *Web Server* 2. Di bagian shell, hasil perhitungan rata-rata program akan ditampilkan, serta lokasi penyimpanan data tersebut. Berikut adalah data hasil dari program pemantauan CPU yang dikumpulkan di *Web Server* 2 dapat dilihat pada tabel 3.

Tabel 3. Hasil Data CPU

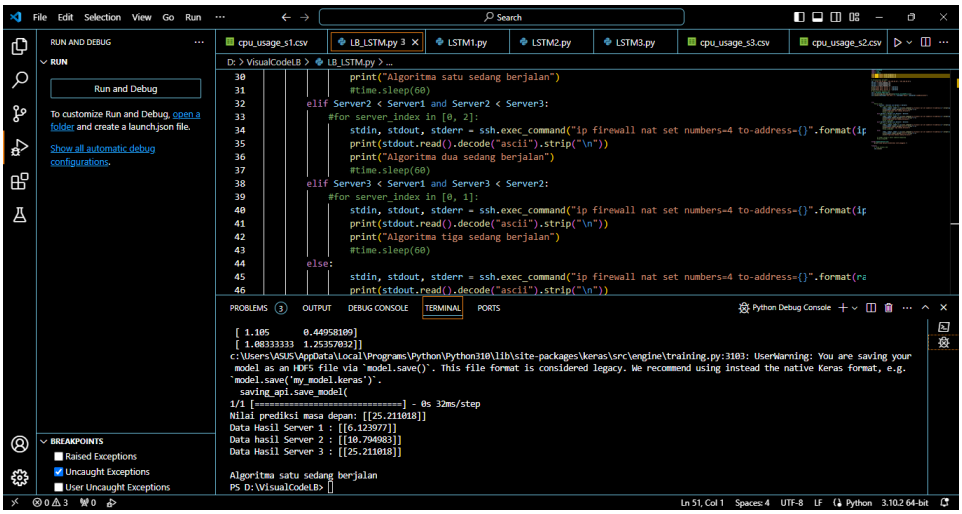
No	*Date*	*CPU Average*
1	07/06/2024 17:30	1.067
2	07/06/2024 17:32	1.008
3	07/06/2024 17:34	1.072
4	07/06/2024 17:36	1.076
5	07/06/2024 17:38	1.044
:	:	:
56	10/06/2024 13:14	1.904
57	10/06/2024 13:16	1.953
58	10/06/2024 13:18	1.942
59	10/06/2024 13:20	1.835
60	10/06/2024 13:22	1.907

Berikut merupakan hasil visualisasi data rata -rata penggunaan *CPU* berbentuk grafik dengan lama pengumpulan data ditampilkan selama 1 jam, di masing-masing *web server*.

p-ISSN: 2775-4227
e-ISSN: 2775-4219

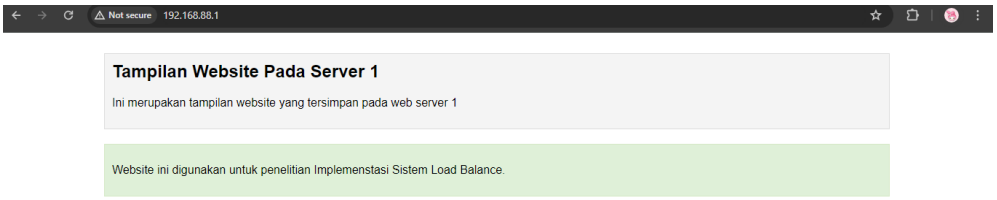
2. Pengujian dan Hasil *Load Balancing*

Pada tahap ini, penelitian menguji skenario algoritma yang telah dibuat dengan menambah beban permintaan *(request)* ke web server di alamat IP 192.168.88.1 menggunakan perangkat lunak *Apache JMeter* dan *JavaScript*. Hasilnya kemudian diperiksa melalui browser pada klien untuk memastikan apakah algoritma berjalan sesuai dengan rancangan seperti pada gambar 8.



Gambar 11. Pengujian *Load Balancing*

Pada Gambar 10 pengujian *load balancing* mendapatkan nilai hasil prediksi LSTM tersebut, terdapat data hasil server 1: [[6.123977]], data hasil server 2: [[10.794983]], dan data hasil server 3: [[25.211018]]. Prediksi ini menunjukkan penggunaan rata-rata CPU untuk masing-masing server berdasarkan model LSTM yang telah dilatih. Data ini kemudian dapat digunakan untuk mengevaluasi performa sistem dan membuat keputusan terkait manajemen beban server. Evaluasi lebih lanjut dapat dilakukan dengan membandingkan hasil prediksi ini dengan data aktual untuk mengukur akurasi dan efektivitas model LSTM dalam memprediksi penggunaan CPU.



Gambar 12. Hasil *Load Balancing*

Gambar 12 Hasil *Load Balancing* menunjukkan bahwa halaman web ini adalah bagian dari pengujian implementasi sistem load balancing yang dilakukan pada jaringan lokal. Dengan mengakses IP jaringan server 192.168.88.1, pengguna dapat mengidentifikasi bahwa mereka sedang melihat konten yang di-*host* oleh Server 1. Hal ini membantu dalam penelitian dan pengujian untuk memastikan bahwa load balancer berfungsi dengan baik dalam mendistribusikan beban ke berbagai server dan mengoptimalkan kinerja keseluruhan sistem.

3. Hasil Metode LSTM

Pada tahap ini, dilakukan pengujian akurasi metode LSTM dalam memprediksi data. Saat pengujian, data yang sebelumnya dibagi menjadi data latih dan data tes akan dihitung menggunakan metrik *Root Mean Squared Error* (RMSE), *Mean Absolute Error* (MAE), dan *Mean Absolute Percentage Error* (MAPE). Data yang digunakan untuk pengujian akurasi model LSTM adalah data rata-rata penggunaan CPU setiap 1 jam. Berikut tabel 4 hasil LSTM.

Tabel 4. Hasil LSTM

No	Pengujian	Server 1	Server 2	Server 3
1	RMSE	0.757	1.909	1.033
2	MAE	0.656	1.526	0.860
3	MAPE	8.059	23.003	29.106

p-ISSN: 2775-4227
e-ISSN: 2775-4219

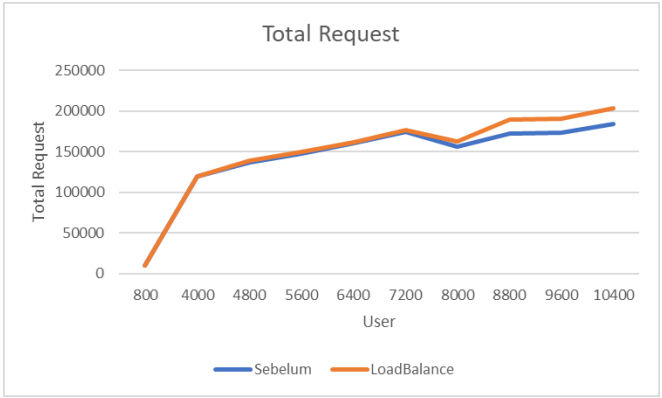
4. Pengujian Kinerja Sistem *Load Balancing Web Server*

Pengujian dilakukan sebanyak dua kali menggunakan perangkat lunak *Apache Jmeter*. Percobaan pertama dilakukan sebelum sistem *load balancing* diaktifkan, percobaan kedua saat program *load balancing* berjalan dan sedang melakukan perpindahan server. Setiap percobaan dilakukan sebanyak 11 kali dengan skenario yang sama namun dengan jumlah pengguna yang berbeda-beda. Pada tabel 5 terlihat bahwa sebelum, dan saat sistem *load balancing* dijalankan, performa sistem menunjukkan variasi *users*, total *request*, waktu respons rata-rata, *throughput*, dan *persentase error*. Berikut merupakan tabel pengujian kinerja sistem *load balancing* dapat dilihat pada Tabel 5.

Tabel 5. Pengujian Kinerja Sistem *Load Balancing* Web Server

No	Status	user	Total Request	Waktu Respons Rata-rata (ms)	Throughput (Request/s)	Error (%)
1	Sebelum	800	20000	13	991,3	0
	sesudah	800	20000	30	945,9	0
2	Sebelum	4000	9954	224	3366	0,33
	Load Balance	4000	10000	66	4357,3	0
3	Sebelum	4800	120000	138	4319,5	0
	Load Balance	4800	120000	108	5008,6	0
4	Sebelum	5600	137312	233	3829,6	0,08
	Load Balance	5600	138752	202	4096,1	0,04
5	Sebelum	6400	147208	244	4135,4	0,36
	Load Balance	6400	149392	320	4281,4	0,3
6	Sebelum	7200	160560	381	4193,4	0,5
	Load Balance	7200	161806	369	4228,8	0,47
7	Sebelum	8000	174200	401	4270,2	0,62
	Load Balance	8000	176800	389	4497,4	0,33
8	Sebelum	8800	156640	717	3665,6	1,69
	Load Balance	8800	162448	661	4093,4	1,48
9	Sebelum	9600	172500	689	4231,2	1,83
	Load Balance	9600	190032	616	5141,4	1,1
10	Sebelum	10400	173919	646	4292,8	2,06
	Load Balance	10400	190400	619	5300,1	1,27
11	Sebelum	11200	184168	797	4091,1	2,17
	Load Balance	11200	203752	715	5017,7	1,56
Peningkatan Rata-rata (%)			**4%**	**9%**	**12%**	**32%**

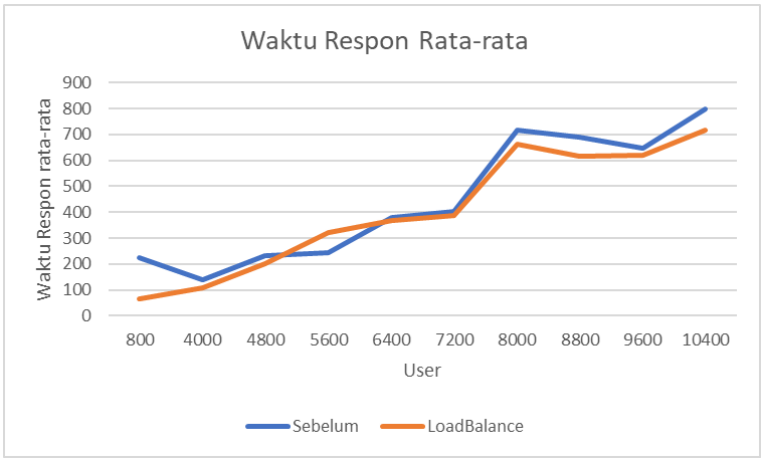
Pada tabel 5 menunjukkan peningkatan total permintaan sebesar 4% setelah sistem *load balancing* berbasis LSTM diterapkan. Peningkatan ini menunjukan bahwa metode prediksi berbasis LSTM dapat mendistribusikan beban secara lebih merata dibandingkan algoritma tradisional seperti Round Robin. Waktu respons rata-rata meningkat sebesar 9%, yang dapat diartikan sebagai peningkatan efisiensi. *Throughput* meningkat sebesar 12%, menunjukkan bahwa sistem dapat menangani lebih banyak permintaan per detik setelah *load balancing*. Selain itu, penurunan rata-rata kesalahan sebesar 32% menunjukkan peningkatan keandalan sistem setelah load balancing diterapkan.



Gambar 13. Grafik Total Request

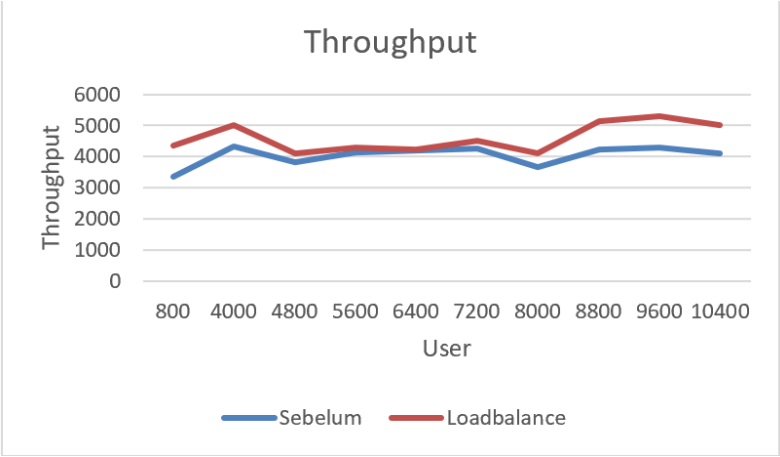
p-ISSN: 2775-4227
e-ISSN: 2775-4219

Gambar 13 menunjukkan total permintaan (Total *Request*) dalam dua skenario: "Sebelum" (garis biru) dan *"LoadBalance"* (garis oranye), dengan jumlah pengguna *(User)* dari 800 hingga 10400. Pada skenario "Sebelum," total permintaan meningkat stabil, mencapai sekitar 150.000. Sementara itu, *"LoadBalance"* menunjukkan peningkatan cepat dan lebih tinggi, mencapai sekitar 200.000. Secara keseluruhan, *"LoadBalance"* menghasilkan permintaan lebih tinggi, menunjukkan efektivitas *load balancing* dalam menangani peningkatan permintaan.



Gambar 11. Grafik Waktu Respon Rata-rata

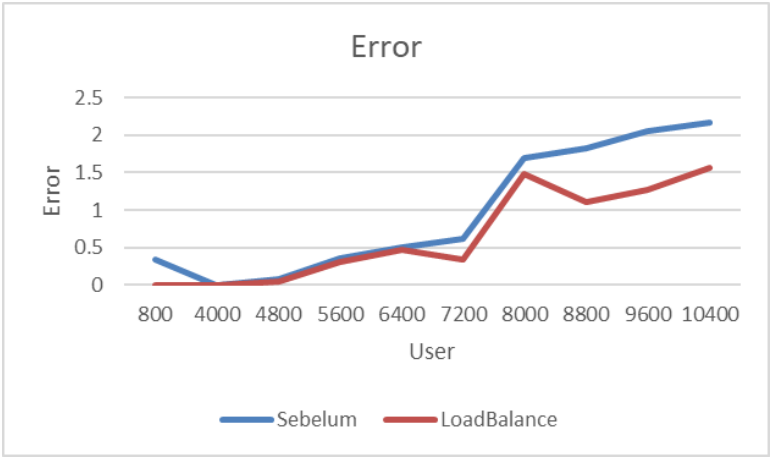
Gambar 11 menunjukkan waktu respons rata-rata dalam dua kondisi: "Sebelum" (garis biru) dan "LoadBalance" (garis oranye), dengan jumlah pengguna dari 800 hingga 10400. Pada kondisi "Sebelum," waktu respons meningkat dengan bertambahnya pengguna, mencapai sekitar 850 ms pada 10400 pengguna. Sebaliknya, kondisi *"LoadBalance"* menunjukkan peningkatan yang lebih lambat dan stabil, dengan puncak sekitar 750 ms. Rata-rata, waktu respons pada kondisi *"LoadBalance"* lebih rendah, menunjukkan efektivitas load balancing dalam mengurangi waktu respons rata-rata saat jumlah pengguna meningkat.



Gambar 14. Grafik *Throughput*

Gambar 12 menunjukkan *throughput* dalam dua kondisi, yaitu "Sebelum" (garis biru) dan *"Loadbalance"* (garis oranye), dengan jumlah pengguna dari 800 hingga 10400. Pada kondisi "Sebelum," *throughput* stabil di sekitar 4000 dengan sedikit fluktuasi. Sebaliknya, kondisi *"Loadbalance"* menunjukkan *throughput* yang lebih tinggi dan stabil di kisaran 4500 hingga 5000, dengan beberapa puncak di atas 5000. Rata-rata, kondisi *"Loadbalance"* menunjukkan *throughput* yang lebih tinggi dibandingkan kondisi "Sebelum," menunjukkan bahwa penggunaan load balancing meningkatkan efisiensi sistem dalam menangani jumlah pengguna yang lebih besar.

p-ISSN: 2775-4227
e-ISSN: 2775-4219



Gambar 15. Grafik Error

Gambar 15 menunjukkan error antara metode "Sebelum" dan *"LoadBalance"* untuk jumlah pengguna dari 800 hingga 10400. Metode "Sebelum" menunjukkan error rata-rata lebih tinggi dibandingkan *"LoadBalance"* di sebagian besar titik data. Kedua metode memiliki error rendah dan stabil hingga 4800 pengguna, tetapi meningkat signifikan setelah 5600 pengguna. Metode *"LoadBalance"* memiliki error lebih rendah, terutama antara 7200 dan 8800 pengguna. Secara keseluruhan, *"LoadBalance"* lebih efektif dalam mengurangi error dibandingkan "Sebelum".

4. KESIMPULAN

Berdasarkan hasil pengujian dan pembahasan, implementasi ini berhasil meningkatkan kinerja web server dengan menyeimbangkan beban dan mengurangi risiko kelebihan beban. Pengujian menunjukkan peningkatan total permintaan sebesar 4% (dari 143.646 menjadi 150.338), penurunan waktu respons rata-rata sebesar 9% (dari 447 ms menjadi 406,5 ms), penurunan tingkat error sebesar 32% (dari 0,964 menjadi 0,655), serta peningkatan throughput sebesar 12% (dari 4039,48 menjadi 4602,22). Metode LSTM diterapkan untuk memprediksi beban CPU server dengan mengukur kinerja menggunakan RMSE, MAE, dan MAPE. Hasil ini membuktikan bahwa metode LSTM efektif untuk load balancing, sehingga layanan web menjadi lebih cepat dan andal. Penelitian ini memberikan kontribusi penting dalam mengoptimalkan distribusi beban server web berbasis prediksi dan dapat menjadi alternatif inovatif dalam pengembangan sistem load balancing. Penelitian mendatang disarankan untuk menguji metode ini pada skala server yang lebih besar atau mengkombinasikannya dengan algoritma lain untuk hasil yang lebih optimal. Untuk penelitian selanjutnya, disarankan untuk mengeksplorasi metode prediksi lain seperti *Gated Recurrent Unit* (GRU) atau mengintegrasikan algoritma LSTM dengan teknik *load balancing* adaptif. Penelitian ini juga perlu diuji pada server dengan infrastruktur yang lebih kompleks untuk mengevaluasi performanya dalam skenario produksi nyata.

DAFTAR PUSTAKA

[1] M. Waluyo, F. Antony, and C. Setiawan, "Implementasi Load Balancing Web Server Dengan Haproxy Menggunakan Algoritma Round Robin," *Journal of Intelligent Networks and IoT Global*, vol. 1, pp. 46–52, Jul. 2023, doi: 10.36982/jinig.v1i1.3074.

[2] E. P. Cynthia, I. Iskandar, and A. A. Sipayung, "Rancang Bangun Server HAproxy Load Balancing Master to Master MySQL (Replication) Berbasis Cloud Computing," *Algoritma : Jurnal Ilmu Komputer Dan Informatika*, vol. 4, no. 1, pp. 45–54, 2020, doi: 10.30829/algoritma.v4i1.7275.

[3] A. Fadila and M. Nasir, "JAISE : Journal of Artificial Intelligence and Software Engineering Implementasi Sistem Load Balancing Web server Pada Jaringan Public Cloud Computing Menggunakan Least Connection."

[4] S. D. Riskiono and D. Pasha, "Analisis Perbandingan Server Load Balancing dengan Haproxy & Nginx dalam Mendukung Kinerja Server E- Learning," *Jurnal Telekomunikasi dan Komputer*, vol. 10, no. 3, p. 135, 2020, doi: 10.22441/incomtech.v10i3.8751.

[5] Y. Permana, R. Ritzkal, and Y. Afrianto, "Load Balancing Method Performance Analysis on Haproxy and Router OS," *Jurnal Mantik*, vol. 4, no. 3, pp. 1588–1596, 2020, [Online]. Available: https://iocscience.org/ejournal/index.php/mantik

p-ISSN: 2775-4227
e-ISSN: 2775-4219

[6] T. Wira Harjanti, H. Setiyani, and J. Trianto, "Load Balancing Analysis Using Round-Robin and Least-Connection Algorithms for Server Service Response Time," *Applied Technology and Computing Science Journal*, vol. 5, no. 2, pp. 40–49, 2022, doi: 10.33086/atcsj.v5i2.3743.

[7] M. E. Mumcuoglu *et al.*, "Driving behavior classification using long short term memory networks," *2019 AEIT International Conference of Electrical and Electronic Technologies for Automotive, AEIT AUTOMOTIVE 2019*, 2019, doi: 10.23919/EETA.2019.8804534.

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

[9] M. A. Nugroho and R. Kartadie, "Analisis Kinerja Penerapan Container untuk Load Balancing Web Server," *JIPI (Jurnal Ilmiah Penelitian dan Pembelajaran Informatika)*, vol. 1, no. 02, pp. 7–15, 2016, doi: 10.29100/jipi.v1i02.35.

[10] B. Tjahjono, A. Sulaeman, F. Adikara, and K. Juman, "Implementation of Load Balancing Technology Using Raspberry Pi as a Server for Computer Based Examination," *International Conference on Islam, Science and Technology*, 2020, doi: 10.4108/eai.2-10-2018.2295570.

[11] A. Hanafiah and R. Wandri, "Implementasi Load Balancing Dengan Algoritma Penjadwalan Weighted Round Robin Dalam Mengatasi Beban Webserver," *IT Journal Research and Development*, vol. 5, no. 2, pp. 226–233, 2021, doi: 10.25299/itjrd.2021.vol5(2).5795.

[12] A. A. Aulia, A. M. Elhanafi, H. Dafitri, A. Aulia, A. M. Elhanafi, and H. Dafitri, "Implementasi Algoritma Gated Recurrent Unit Dalam Melakukan Prediksi Harga Kelapa Sawit Dengan Memanfaatkan Model Recurrent Neural Network (RNN)," *Prosiding SNASTIKOM: Seminar Nasional Teknologi Informasi & Komunikasi Paper*, pp. 288–294, 2021.

[13] L. Wang, L. Bai, Z. Li, R. Zhao, and F. Tsung, "Correlated Time Series Self-Supervised Representation Learning via Spatiotemporal Bootstrapping," *IEEE International Conference on Automation Science and Engineering*, vol. 2023-Augus, 2023, doi: 10.1109/CASE56687.2023.10260640.

[14] T. O. Hodson, "Root-mean-square error (RMSE) or mean absolute error (MAE): when to use them or not," *Geosci Model Dev*, vol. 15, no. 14, pp. 5481–5487, 2022, doi: 10.5194/gmd-15-5481-2022.

[15] S. Prayudani, A. Hizriadi, Y. Y. Lase, Y. Fatmi, and Al-Khowarizmi, "Analysis Accuracy of Forecasting Measurement Technique on Random K-Nearest Neighbor (RKNN) Using MAPE and MSE," *J Phys Conf Ser*, vol. 1361, no. 1, 2019, doi: 10.1088/1742-6596/1361/1/012089.

[16] M. Dartono and D. Irawan, "Penerapan Metode Per Connection Classifier (PCC) Pada Perancangan Load Balancing Dengan Router Penerapan Metode Per Connection Classifier (Pcc) Pada Perancangan Load Balancing Dengan Router Mikrotik."

[17] N. Agung Pambudi and A. Hendri Hendrawan, "Digital Information Board Using Rasberry Pi 3 Model B Based on Raspbian," *Jurnal Mantik*, vol. 4, no. 4, pp. 2588–2592, 2021, [Online]. Available: https://iocscience.org/ejournal/index.php/mantik

[18] S. K. Rout, J. V. R. Ravindra, A. Meda, S. N. Mohanty, and V. Kavididevi, "A Dynamic Scalable Auto-Scaling Model as a Load Balancer in the Cloud Computing Environment," *EAI Endorsed Transactions on Scalable Information Systems*, vol. 10, no. 5, pp. 1–7, 2023, doi: 10.4108/eetsis.3356.