Jurnal Pendidikan dan Teknologi Indonesia (JPTI)
Vol. 4, No. 12, Desember 2024, Hal. 669-677
DOI: https://doi.org/10.52436/1.jpti.512
p-ISSN: 2775-4227
e-ISSN: 2775-4219

Implementasi Sistem Load Balancing Web Server Menggunakan Metode Auto Regressive Integrated Moving Average (ARIMA)

Brisman Armansyah Sulaikandi[*1], Yuggo Afrianto[2] , Bayu Adhi Prakosa[3]

[1,2,3]Univesitas Ibn Khaldun Bogor, Indonesia
Email: [1]brismanarmansyah.s@gmail.com, [2]yuggo@uika-bogor.ac.id, [3]bayu.adhi@uika-bogor.ac.id

Abstrak

Web server adalah perangkat lunak yang menyediakan layanan berbasis data dan merespons permintaan dari *web browser*, Akan tetapi masalah umum yang dihadapi oleh *server web*, seperti lalu lintas tinggi, waktu henti, beban yang tidak merata, dan pemilihan *server* yang tidak tepat dapat mengurangi kinerja dan efiensi sistem, masalah tersebut juga dapat diatasi secara efektif melalui berbagai teknik *Load Balance*. Teknik *load balancing* adalah proses mendistribusikan beban kerja di berbagai sumber daya komputasi untuk mengoptimalkan waktu respon dan meningkatkan kinerja sistem. Penelitian ini bertujuan untuk meningkatkan kinerja *web server* melalui implementasi sistem *load balancing* berbasis metode *Auto Regressive Integrated Moving Average* (ARIMA). Teknik ini diterapkan untuk memprediksi penggunaan *Central Processing Unit/Processor* (CPU) dan mengarahkan lalu lintas ke *server* dengan beban paling ringan. Hasil menunjukkan peningkatan *throughput* dan *total request* pada *server* sebesar 27% dan 58%, serta penurunan waktu respon rata-rata dan persentase *error* sekitar 18% dan 46%. Metode ini memberikan kontribusi signifikan terhadap optimasi kinerja sistem dan manajemen sumber daya komputasi, serta dapat mengatasi masalah beban yang tidak merata dan pemilihan *server* yang tidak tepat secara efektif.

Kata kunci: *ARIMA, data time series, load balance, optimasi web server, prediksi penggunaan CPU, web server*

Implementation of Web Server Load Balancing System Using Auto Regressive Integrated Moving Average (ARIMA) Method

Abstract

A web server is software that provides data-driven services and responds to requests from web browsers, However, common problems faced by web servers, such as high traffic, downtime, uneven load, and improper server selection can reduce system performance and efficiency, these problems can also be effectively addressed through various Load Balance techniques. Load balancing technique is the process of distributing workload across various computing resources to optimize response time and improve system performance. This research is conducted to improve web server performance through the implementation of load balancing system based on Auto Regressive Integrated Moving Average (ARIMA) method. This technique is applied to predict Central Processing Unit/Processor (CPU) utilization and direct traffic to the server with the lightest load. Results show an increase in throughput and total requests on the server by 27% and 58%, as well as a decrease in average response time and error percentage by about 18% and 46%. This method makes a significant contribution to system performance optimization and computing resource management, and can effectively solve the problems of uneven load and inappropriate server selection.

Keywords: *ARIMA, CPU usage prediction, data time series, load balancing, web server, web server optimization*

1. PENDAHULUAN

Web Server adalah perangkat lunak server yang menyediakan layanan berbasis data, menerima dan merespons permintaan dari *web browser* [1], layanan yang biasa diterima oleh *web server* protokol *Hypertext Transfer Protocol* (HTTP) akan tetapi *web server* tidak hanya dapat menerima layanan protokol HTTP, tetapi dapat menerima layanan protokol lainnya seperti menerima permintaan layanan protokol *Hypertext Transfer Protocol Secure* (HTTPS), *File Transfer Protocol* (FTP), dan *Simple Mail Transfer Protocol* (SMTP) [2].

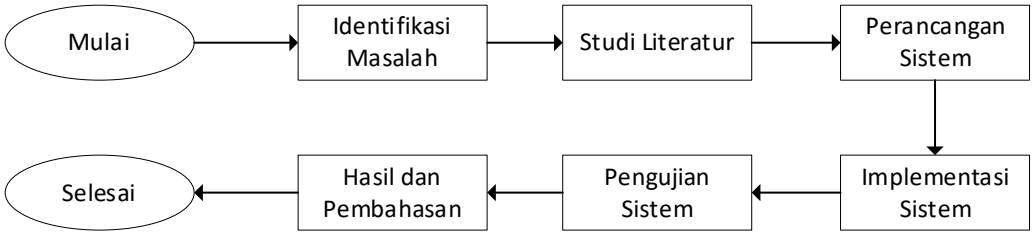
p-ISSN: 2775-4227
e-ISSN: 2775-4219

Akan tetapi masalah umum yang dihadapi oleh *server web*, seperti lalu lintas tinggi, waktu henti, beban yang tidak merata, dan prediksi pemilihan *server* yang tidak tepat, dapat diatasi secara efektif melalui berbagai teknik. Penyeimbangan beban, misalnya, dapat membantu mendistribusikan lalu lintas ke beberapa *server*, mencegah kelebihan beban dan waktu henti [3]. Seperti dalam penerapan *load balancing* berbagai skenario dunia nyata di dalam kasus PT GO-JEK Indonesia, penggunaan teknik *load balancing* menggunakan metode *Next Hop (NTH)* dan *failover* dengan dua *Internet Service Provider* (ISP) dan konfigurasi yang tepat dapat meningkatkan kinerja dan keandalan jaringan [4]. Oleh sebab itu penggunaan *load balancing* dapat digunakan. *load balancing* adalah teknik penting dalam manajemen jaringan, memastikan arus lalu lintas yang optimal dan mencegah kelebihan beban pada koneksi tertentu [5].

Pada penelitian ini mencoba untuk menggunakan metode *Auto Regressive Integrated Moving Average* (ARIMA) yaitu salah satu dari model analisis *data time series* yang digunakan untuk memprediksi nilai dari masa yang akan datang meggunakan analisis pada data deret waktu (*time series*) [6], seperti data waktu kinerja beban pada *Server* untuk melakukan analisa pola dan trend dari suatu variabel data sehingga menghasilkan ramalan atau prediksi untuk waktu tertentu. Informasi ini dapat digunakan untuk mengatur dan mendistribusikan sumber daya dengan lebih efisien sesuai dengan permintaan yang diprediksi. Yang dimaksud dengan data *time series* sendiri adalah kumpulan observasi dari satu subjek pada interval waktu yang berbeda [7]. Dengan kata lain, data *time series* adalah data yang memiliki waktu tertentu yang terikat dengannya. Data *time series* yang digunakan dalam penelitian ini adalah data rata-rata penggunaan CPU per menitnya. Penelitian ini bertujuan untuk mengisi kekosongan dalam literatur terkait penerapan ARIMA pada load balancing, serta menawarkan solusi prediksi penggunaan CPU yang dapat meningkatkan efisiensi web server secara signifikan.

2. METODE PENELITIAN

Metode penelitian dibuat dalam bentuk tahapan alur supaya setiap proses dalam penelitian berjalan secara sistematis. Tahapan alur dalam penelitian dapat dilihat pada Gambar 1.



Gambar 1. Metode Penelitian

Pada gambar 1. Dapat dilihat penelitian ini dilakukan dalam lima tahap utama: (1) identifikasi masalah, (2) studi literatur, (3) perancangan sistem, (4) implementasi sistem, dan (5) pengujian sistem. Data penggunaan CPU dikumpulkan menggunakan skrip Python pada masing-masing web server, dan prediksi dilakukan menggunakan model ARIMA yang diimplementasikan pada Raspberry Pi.

2.1. Identifikasi Masalah

Tahapan awal diawali dengan mengidentifikasi masalah sesuai dengan bagian pendahuluan untuk mengetahui tujuan dan manfaat dari dilakukannya penelitian ini.

2.2. Studi Literatur

Tahapan ini dilakukan untuk mempelajari semua aspek yang dibutuhkan dalam penelitian ini, yang pertama yaitu mencari contoh penerapan *load balancing* pada sistem *web server*, mempelajari metode perhitungan analisis Data *time series* terutama perhitungan dengan model *auto regressive integrated moving average* (ARIMA)..

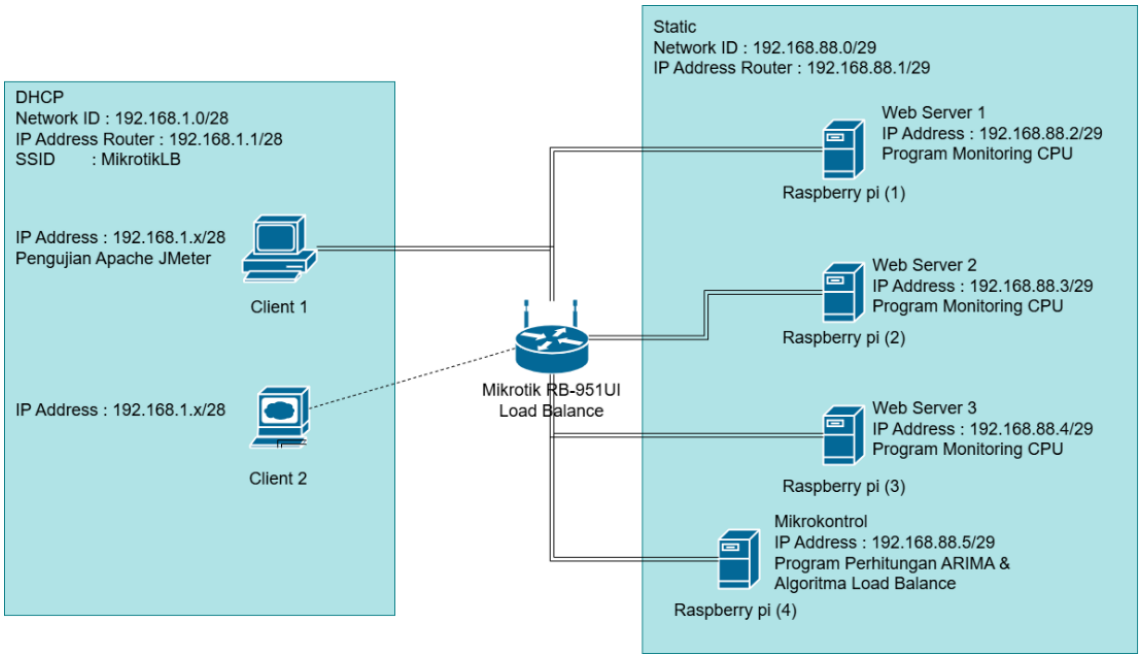
2.3. Perancangan Sistem

Pada tahapan perancangan sistem ini dibuat rancangan sistem penerapan *load balancing web server*, diantaranya adalah topologi jaringan yang digunakan peneliti untuk menerapkan dan menguji sistem yang dibuat dan diagram alur program *load balancing web server* secara kesluruhan.

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

Topologi Jaringan

Jaringan komputer yang digunakan dalam penelitian ini menggunakan jaringan lokal yang dimana peneliti menggunakan *router* mikrotik RB-951UI sebagai router pusat pada jaringan ini dan sebagai menjalankan sistem *load balance.*



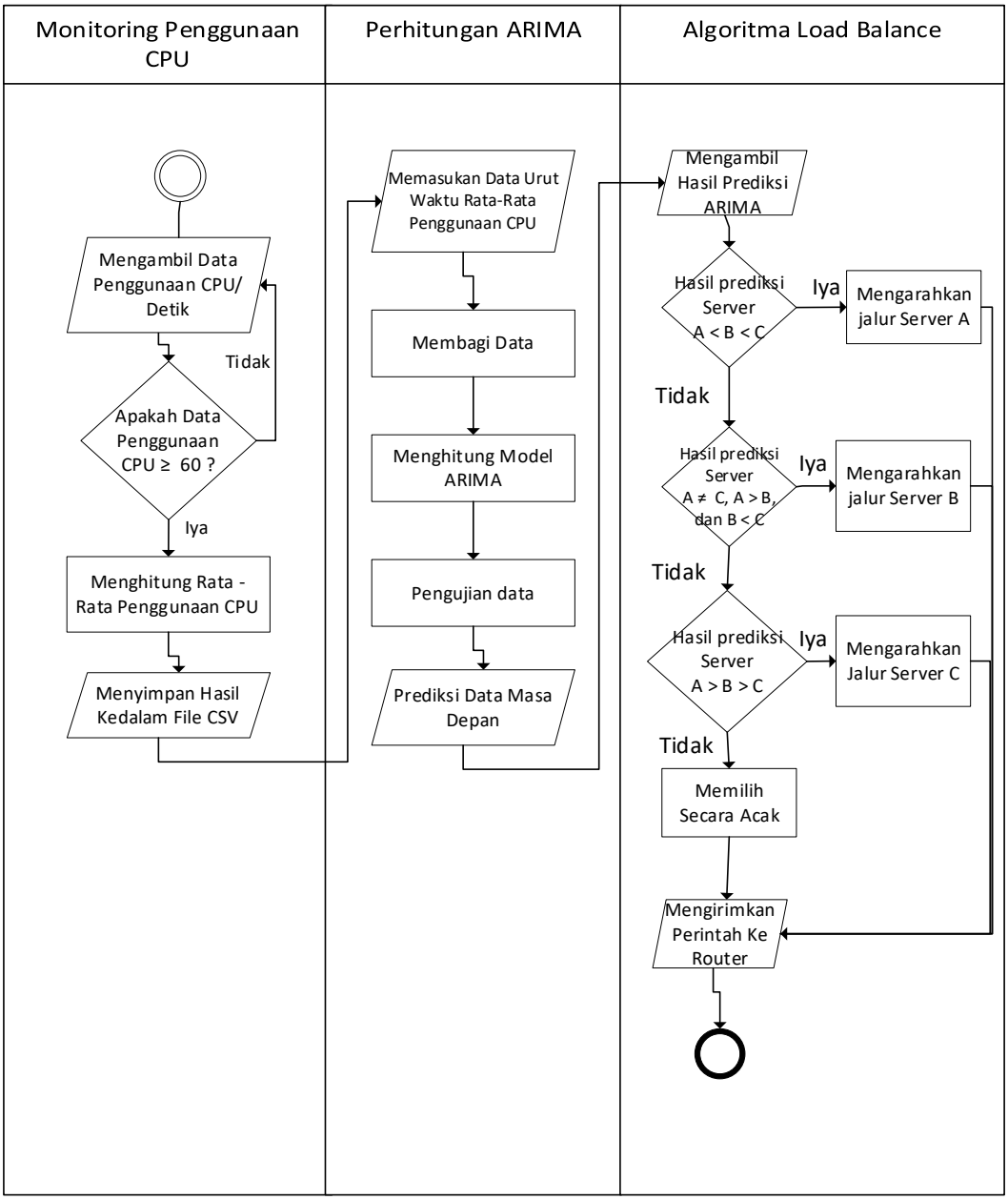
Gambar 2. Topologi Jaringan Logika

Pada gambar 2 dari *router* tersebut dibuat 2 jaringan yang berbeda, diantaranya jaringan *client* yang memiliki *Network* ID 192.168.1.0/28 dengan konfigurasi IP *Dynamic Host Configuration Protocol (DHCP)* dan jaringan *server* dibuat konfigurasi statik dengan *Network* ID 192.168.88.0/29 untuk 4 buah perangkat raspberry pi 3 model B, 3 perangkat raspberry pi digunakan untuk menjalankan *web server* dan menjalankan program pemantauan penggunaan CPU, satu perangkat lainnya digunakan untuk mikrokontroller yang menjalankan program ARIMA dan algoritma *load balancing* yang hasilnya bakal mengirimkan perintah penggantian jalur *load balance* ke *router*. Dalam pengujian sistem *load balancing* menggunakan perangkat lunak Apache Jmeter yang terpasang pada komputer *client* dan terhubung melalui kabel, supaya hasil pengujian yang didapatkan lebih optimal.

Diagram Alur Sistem

Pada penelitian ini membuat 3 program python dengan fungsi yang berbeda-beda, yang pertama program pemantauan penggunaan CPU untuk pengumpulan data, kedua program perhitungan ARIMA untuk mendapatkan hasil prediksi data, dan program ketiga algoritma *load balance* untuk menentukan *server* mana yang akan dipakai dan yang mengirimkan perintah ke router untuk mengganti jalur *web server*.

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



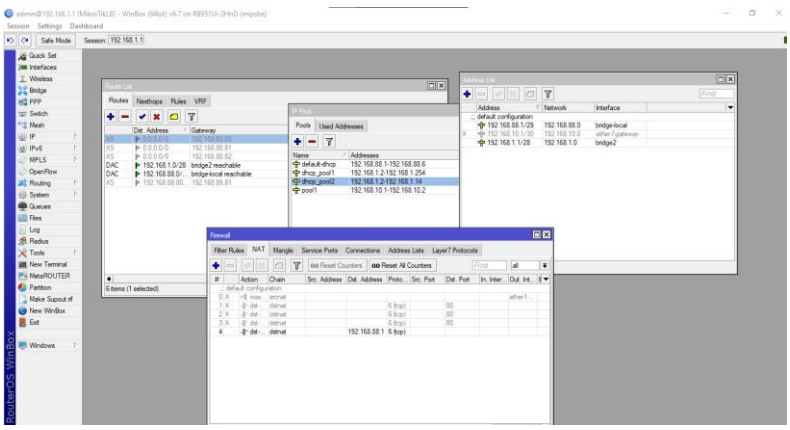
Gambar 3. Diagram Alur Keseluruhan Program Load Balancing

Pada gambar 3. diagram alur keseluruhan program *load balancing*, sistem dimulai dari pengumpalan data rata-tara penggunaan CPU pada masing-masing *web server* selama 1 jam, data yang sudah dikumpulkan dimasukan ke dalam mikrokontroller raspberry menggunakan perintah *curl* pada *command prompt* mikrokontroller raspberry pi atau dapat menggunakan *cronjob* untuk mengambil data secara otomatis, dan data dimasukan ke program perhitungan ARIMA yang mendapatkan hasil prediksi data penggunaan CPU kedepannya, dan mendapatkan pengujian datanya, data hasil prediksi perhitungan ARIMA akan dimasukan ke dalam algoritma *load balance* untuk mencari hasil prediksi terkecil dari masing-masing *web server*, yang nantinya hasil data prediksi terkecil akan dipilih untuk jalur *web server* selanjutnya.

2.4. Implementasi Sistem

Pada tahapan pertama dilakukan penerapan jaringan komputer sesuai pada tahapan rancangan sistem.

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



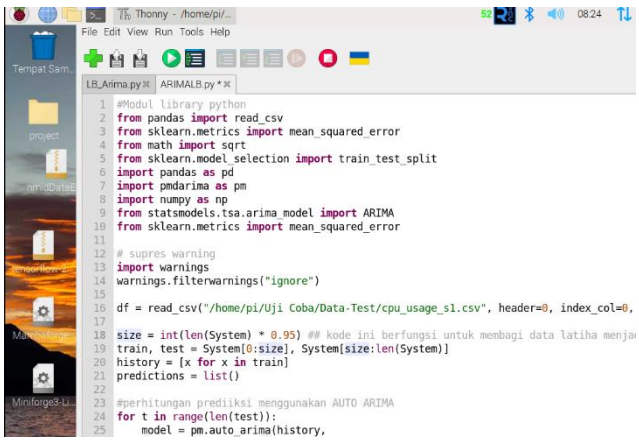
Gambar 4. Konfigurasi Router Mikrotik

Pada gambar 4. pengaturan *router mikrotik dibuat dengan IP address* yang sesuai dengan topologi jaringan sebelumnya ada pada rancangan sistem. Selanjutnya konfigurasi *firewall* NAT DST (*Destination Network Translation*) jaringan yang berfungsi untuk mengarahkan *IP Address router server* menjadi alamat *web server* yang di pilih melalui sistem *load balance* yang dibuat. Selanjutnya dilakukan penerapan dari rancangan sistem yang dibuat.

```
import os
import time

# Nama file CSV untuk menyimpan data
file_name = os.path.expanduser("/var/www/html/cpu_usage_s1.csv")

# Inisialisasi list untuk menyimpan data penggunaan CPU
cpu_usages = []

# Fungsi untuk mengambil data beban CPU
def get_cpu_usage():
    return psutil.cpu_percent(interval=1)

# Fungsi untuk menghitung rata-rata penggunaan CPU
def calculate_average(cpu_usages):
    return sum(cpu_usages) / len(cpu_usages)

# Fungsi untuk membuat file CSV jika belum ada
def create_csv_file_if_not_exists():
    if not os.path.exists(file_name):
        with open(file_name, mode="w", newline="") as file:
            writer = csv.writer(file)
            writer.writerow(["Date", "CPU_Average"])

# Fungsi untuk menghapus baris pertama jika jumlah baris sudah mencapai 60
def manage_csv_file():
    df = pd.read_csv(file_name)
    if len(df) >= 120:
        df = df.iloc[1:]  # Hapus baris pertama
```

Gambar 5. Promgram Monitoring Penggunaan CPU

Pada gambar 5 merupakan gambar pemograman sistem monitoring penggunaan CPU menggunakan bahasa pemograman python yang ditaruh ke *web server*. Untuk pemograman ini dibuat untuk melakukan pengumpulan data sebanyak 60 per detiknya yang akan dihitung rata-rata dari data yang dikumpulkan, data tersebut juga akan dikumpulkan lagi sebanyak 60 menit sebagai output dalam program ini berformat file csv

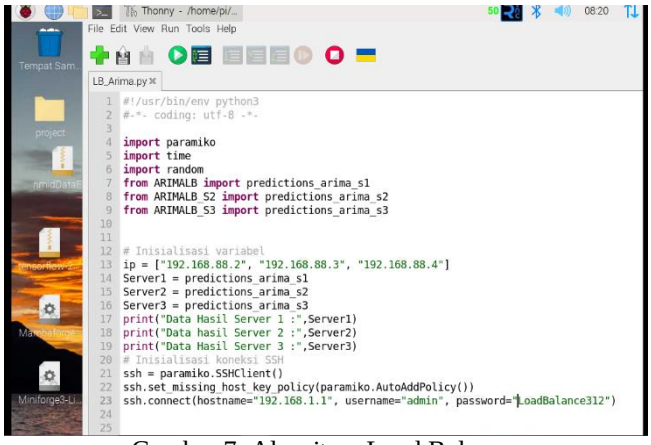


```
#Modul Library python
from pandas import read_csv
from sklearn.metrics import mean_squared_error
from math import sqrt
from sklearn.model_selection import train_test_split
import pandas as pd
import pmdarima as pm
import numpy as np
from statsmodels.tsa.arima_model import ARIMA
from sklearn.metrics import mean_squared_error

# supres warning
import warnings
warnings.filterwarnings("ignore")

df = read_csv("/home/pi/Uji Coba/Data-Test/cpu_usage_s1.csv", header=0, index_col=0,

size = int(len(System) * 0.95) ## kode ini berfungsi untuk membagi data latiha menja
train, test = System[0:size], System[size:len(System)]
history = [x for x in train]
predictions = list()

#perhitungan prediiksi menggunakan AUTO ARIMA
for t in range(len(test)):
    model = pm.auto_arima(history,
```

Gambar 6. Program Perhitungan ARIMA

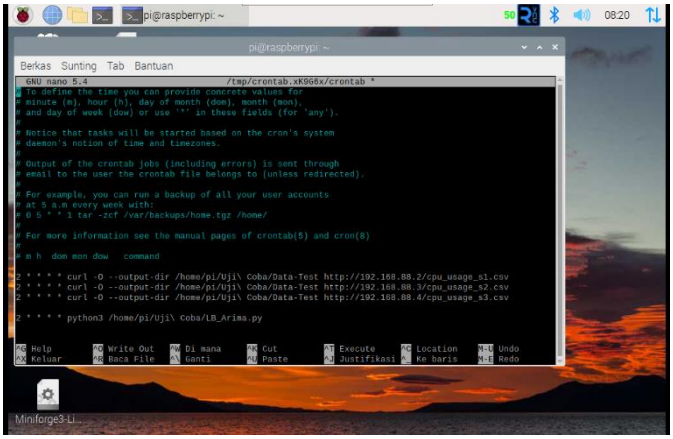
p-ISSN: 2775-4227
e-ISSN: 2775-4219

Pada gambar 6. Merupakan hasil penerapan pemograman ARIMA dengan python yang diletakan pada raspberry mikrokontroler. Program ini memerlukan masukan data *time series* rata-rata penggunaan CPU yang sebelumnya dikumpulkan pada *web server*, data yang dikumpulkan bakal dibagi menjadi data latih dan data tes, selanjutnya dilakukan perhitungan model ARIMA menggunakan data latih,data prediksi yang menggunakan data latih akan dihitung akurasi prediksinya menggunakan matriks *error Mean Absolute Error* (MAE), *Mean squared Error* (MSE), *dan Root Mean Squared Error* (RMSE). Lalu akan diprediksi 1 langkah masa depan yang akan menjadi *output* pada program ini.



Gambar 7. Algoritma Load Balance.

Pada gambar 7 Program algoritma load balance yang telah dibuat mengikuti rancangan sebelumnya dengan input dari program ini merupakan variabel data prediksi pada masing – masing web server. Algoritma ini bakal mengarahkan jalur web server sesuai berdasarkan algoritma yang diatur seperti ini : Jika *server* A lebih kecil dari *server* B dan C, maka mikrokontroler akan mengalihkan jalur menuju ke *server* A. Jika *server* B lebih kecil dari *server* A dan C, maka mikrokontroler akan mengalihkan jalur menuju ke *server* B. Jika *server* C lebih kecil dari *server* B dan A, maka mikrokontroler akan mengalihkan jalur menuju ke *server* C. Jika hasil prediksi antara *server* A, B, dan C ada yang memiliki nilai sama. maka mikrokontroler akan memilih *server* jalur *server* secara acak.



Gambar 8. Crontab

Pada gambar 8 adalah pengaturan *cronjob* yang berfungsi untuk menjalankan program *load balancing* secara otomatis, dan untuk mengambil data rata-rata penggunaan CPU secara otomatis dari *web server* menggunakan perintah *curl -O --output-dir "lokasi penyimpanan data mikrokontoller" http://alamat-web-server/data_cpu.csv*. Penelitian ini menjalankan program *load balance* setiap 5 menit, dikarenakan, lamanya perhitungan ARIMA dan pengarahan jalur pada mikrokontroller ke router.

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

2.5. Pengujian Sistem

Dalam penelitian ini terdapat 2 data hasil pengujian, yang pertama data pengujian perhitungan ARIMA menggunakan 3 metrik *error*, diantaranya:

1) *Mean Absolute Error* (MAE).

Dalam statistika *Mean Absolute Error* (MAE) adalah ukuran kesalahan antara observasi berpasangan yang mengungkapkan fenomena yang sama. Perhitungan MAE dalam pengujian prediksi time series yaitu rata-rata kesalahan absolut antara nilai data yang diprediksi dengan nilai data aktual [8].

2) Mean Squared Error (MSE).

Mean squared Error (MSE*)* adalah ukuran kesalahan antara observasi berpasangan yang mengungkapkan fenomena yang sama. Perhitungan MSE dalam pengujian prediksi *time serie*s yaitu rata-rata kesalahan kuadrat antara nilai data yang diprediksi dengan nilai data aktual [9].

3) Root Mean Squared Error (RMSE)

Root Mean Squared Error (RMSE) adalah ukuran kesalahan antara observasi berpasangan yang mengungkapkan fenomena yang sama. dalam perhitungan RMSE dalam pengujian prediksi *time series* yaitu dihitung sebagai akar kuadrat dari rata-rata *kuadrat* selisih antara nilai aktual dan nilai prediksi [10].

Dan data hasil pengujian kedua merupakan data pengujian *load balancing* menggunakan perangkat lunak *Apache Jmeter*, untuk mengetahui perbandingan antara sebelum dan saat *load balancing*, data hasil yang didapatkan adalah waktu respon rata-rata, total *request* yang diterima, *throughput*, dan *error* yang terjadi pada saat pengiriman *request*.

3. HASIL DAN PEMBAHASAN

Penelitian ini menggunakan perangkat lunak Apache Jmeter yang digunakan sebagai penambah beban dan sekaligus untuk mengetahui waktu respon rata-rata, total *request*, *throughput* dan *error* yang terjadi pada *web server*.

3.1. Pengujian Prediksi Data Penggunaan CPU

Melakukan pengujian akurasi model *ARIMA* dalam melakukan memprediksi data rata-rata penggunaan CPU. Pada saat pengujian ini data yang sebelumnya dibagi menjadi data latih dan data tes akan dilalukan perhitungan matriks *Root Mean Squared Error (RMSE), Mean Squared Error (MSE), Mean Absolute Error (MAE).* Data yang digunakan untuk pengujian akurasi model *ARIMA* ini data rata-rata penggunaan *CPU* setiap menitnya dalam pengumpulan data selama 30 menit dan 1 jam.

Tabel 1. Hasil Pengujian Model ARIMA

No	Pengujian	Server 1		Server 2		Server 3	
		30 Menit	60 Menit	30 Menit	60 Menit	30 Menit	60 Menit
1	RMSE	44.074	42,784	13.913	41.102	15.514	25.140
2	MSE	1942,506	1830,502	193.571	1689.407	240.693	632.019
3	MAE	380.939	42.784	13.234	36.088	12.379	19.410

Pada tabel 1 hasil pengujian model ARIMA untuk prediksi data penggunaan CPU menunjukan data pengujian dengan data yang dikumpulkan selama 30 menit memiliki akurasi yang lebih baik dari pada data yang dikumpulkan selama 60 menit. Akan tetapi kedua data tersebut memiliki akurasi cukup baik pada matriks RMSE dan MAE, dibandingkan pada matriks MSE.

3.2. Pengujian Sistem Load Balance

Pengujjan sistem *load balance* dilakukan sebanyak 10 kali dengan status pengujian sebelum dan pada saat *load balance* menggunakan perangkat lunak *apache jmeter,* dan memiliki total **user** yang berbeda-beda setiap percobaannya, dimulai dengan 4000 user ditambah 800 *user* setiap percobaan hingga 11200 *user* dipercobaan ke-10.

Tabel 2. Hasil Pengujian Kinerja Load Balance

No	Status	User	Total Request	Waktu Respon Rata-rata *(ms)*	Throughput *(request/s)*	Error (%)
1.	Sebelum	4000	10000	165	3483,7	0
	Load Balance	4000	10000	111	4050,7	0

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

2.	Sebelum	4800	119040	220	3794,5	0,03
	Load Balance	4800	120000	122	4284,9	0
3.	Sebelum	5600	139664	173	4093,4	0,01
	Load Balance	5600	140000	116	4933,7	0
4.	Sebelum	6400	146418	356	4013,7	0,39
	Load Balance	6400	158728	218	4903,9	0,03
5.	Sebelum	7200	151660	470	3874,1	0,78
	Load Balance	7200	173184	261	5035,9	0,35
6.	Sebelum	8000	165778	484	3994,3	0,86
	Load Balance	8000	184304	327	4727,9	0,35
7.	Sebelum	8800	161608	665	4108,6	1,51
	Load Balance	8800	190720	515	5660,9	0,64
8.	Sebelum	9600	161040	797	3778,8	2,04
	Load Balance	9600	197472	597	5437,5	0,9
9.	Sebelum	10400	177392	765	4116,6	1,94
	Load Balance	10400	206938	571	5190,9	1,07
10.	Sebelum	11200	173832	907	3824,6	2,55
	Load Balance	11200	212176	631	5302,7	1,33
Peningkatan Rata-rata (%)			27%	-18%	58%	-46%

Pada tabel 2. hasil pengujian terakhir kinerja *load balance* dapat dilihat pada 11200 *user* adanya penambahan total *request* pada saat sebelum dan sesudah load balance sekitar 22%, waktu respon rata-rata *web server* menurun sekitar 30%, permintaan per detiknya *(throughput)* menambah 38%, dan permintaan yang gagal menurun sekitar 47%..

4. KESIMPULAN

Berdasarkan hasil penelitian ini penerapan sistem *load balancing web server* dapat disimpulkan bahwa, penerapan *load balance web server* dengan metode ARIMA metode tersebut berhasil sebagai penentu jalur *web server* selanjutnya berdasarkan hasil prediksi nilai masa depan rata-rata penggunaan CPU. dengan hasil uji coba kinerja *load balancing* menggunakan *Apache JMeter* dapat meningkatkan *throughput* dan total *request* sekitar 27% dan 58%, serta dapat menurunkan waktu respon rata-rata dan persentase *error* sekitar 18% dan 46%. Penggunaan model ARIMA dalam prediksi data rata-rata penggunaan CPU dapat dilakukan dengan pengujian metriks error di pengumpulan data sebanyak 60 menit diangka sebagai berikut: *server* 1 dengan *Root Mean Square Error* (RMSE) : 42,784, *Mean Squared Error* (MSE): 1830,502, dan *Mean Absolute Error* (MAE): 42.784; *server* 2 dengan RMSE: 41.102, MSE: 1689.407, dan MAE: 36.088; *server* 3 dengan RMSE: 25.140, MSE: 632.019, dan MAE: 19.410.

DAFTAR PUSTAKA

[1] H. Nurwarsito and V. B. Sejahtera, "Implementation of Dynamic Web Server Based on Operating System-Level Virtualization using Docker Stack," in *2020 12th International Conference on Information Technology and Electrical Engineering (ICITEE)*, IEEE, Oct. 2020, pp. 33–38. doi: 10.1109/ICITEE49829.2020.9271710.

[2] M. M. Vikor, M. I. Kurniansyah, and S. Sinurat, "Sistem Pendukung Keputusan Pemilihan Server Hosting Dan Domain Terbaik Untuk WEB Server Menerapkan Metode VIKOR," 2020. doi: 10.30865/JSON.V2I1.2450.

[3] D. K. Hakim, D. Y. Yulianto, and A. Fauzan, "Pengujian Algoritma Load Balancing pada Web Server Menggunakan NGINX," *JRST (Jurnal Riset Sains dan Teknologi)*, vol. 3, no. 2, p. 85, Sep. 2019, doi: 10.30595/jrst.v3i2.5165.

[4] D. R. Achmmad Mustofa, "Implementasi Load Balancing Dan Failover To Device Mikrotik Router Menggunakan Metode Nth (Studi Kasus: Pt. Go-Jek Indonesia)," *Jurnal Teknologi Informasi dan Ilmu Komputer*, vol. 7, pp. 139–144, 2019, doi: 10.25126/jtiik.202071638.

[5] F. Apriliansyah, I. Fitri, and A. Iskandar, "Implementasi Load Balancing Pada Web Server Menggunakan Nginx," *Jurnal Teknologi dan Manajemen Informatika*, vol. 6, no. 1, pp. 18–26, Apr. 2020, doi: 10.26905/jtmi.v6i1.3792.

[6] C. Zhang and X. Zhou, "Forecasting value-at-risk of crude oil futures using a hybrid ARIMA-SVR-POT

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

model," *Heliyon*, vol. 10, no. 1, p. e23358, Jan. 2024, doi: 10.1016/j.heliyon.2023.e23358.

[7] R. Hyndman, A. Koehler, K. Ord, and R. Snyder, *Forecasting with Exponential Smoothing*. in Springer Series in Statistics. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008. doi: 10.1007/978-3-540-71918-2.

[8] A. A. Suryanto, "PENERAPAN METODE MEAN ABSOLUTE ERROR (MEA) DALAM ALGORITMA REGRESI LINEAR UNTUK PREDIKSI PRODUKSI PADI," *SAINTEKBU*, vol. 11, no. 1, pp. 78–83, Feb. 2019, doi: 10.32764/saintekbu.v11i1.298.

[9] C. Kosma, G. Nikolentzos, N. Xu, and M. Vazirgiannis, "Time Series Forecasting Models Copy the Past: How to Mitigate," *International Conference on Artificial Neural Networks*, pp. 366–378, Jul. 2022, [Online]. Available: http://arxiv.org/abs/2207.13441

[10] M. Kang and J. Yu, "Multi-step Forecast of Ending Balance Based on Machine Learning," in *2020 IEEE 5th International Conference on Cloud Computing and Big Data Analytics (ICCCBDA)*, IEEE, Apr. 2020, pp. 206–210. doi: 10.1109/ICCCBDA49378.2020.9095752.