PERFORMANCE ANALYSIS OF LOAD BALANCING LEARNING MANAGEMENT SYSTEM MOODLE ON DOCKER CONTAINER

Bayu Setyaji[1], Mega Pranata[2], Iqsahiro Kresna[3]
Institut Teknologi Telkom Purwokerto[123]

E-mail : 19102157@ittelkom-pwt.ac.id

ABSTRAK

Permintaan yang berlebih pada *web server* menyebabkan *server* tidak mampu melayani permintaan yang mengakibatkan *overload* dan menyebabkan *server down*. *Load balancing* dapat menjadi solusi, dengan kemampuan membagi beban kerja ke beberapa *server* secara seimbang. Metode *load balancing* dapat diterapkan pada teknologi *container.* Docker merupakan salah satu teknologi *container* yang berfungsi menjalankan aplikasi dengan cepat dan terisolasi didalam sebuah wadah yang disebut *container*.
Penelitian ini ditujukan untuk mengukur performa dari metode *load balancing* untuk server *learning management system* dengan dua algoritma yaitu *least connection* dan *round robin* yang diterapkan dalam *container docker*. Pengujian dilakukan dengan memberikan *request* pada *server* dengan tiga variasi yaitu 1000, 2000 dan 4000 *request*. Hasil pengujian mendapatkan hasil bahwa algoritma least connection mendapatkan hasil yang lebih baik pada pengujian dengan request 1000 dan 2000 dengan nilai throughput tertinggi adalah 4,08333 Mbps. Kemudian nilai *response time* algoritma *least connection* unggul pada pengujian ke 1000 dan 2000 sedangkan *round robin* unggul pada request 4000 dengan nilai 4985,31 ms. Lalu pada pengujian nilai *error* algoritma *least connection* memiliki nilai *error* lebih rendah pada semua skema pengujian dan pada penggunaan CPU.

Kata kunci : Container, docker, least connection, learning management system, round robin.

1. PENDAHULUAN

Penggunaan internet di Indonesia semakin meningkat dari tahun ke tahun, terutama pada bidang Pendidikan. Berdasarkan data dari Badan Pusat Statistik tahun 2016 – 2020 penggunaan internet sebagai sarana pembelajaran pada rentang usia 5 – 24 tahun meningkat sebesar 25,35% [1]. Pesatnya pertumbuhan penggunaan internet dalam saranan pembelajaran harus menjadu perhatian yang serius, khusus nya pada layanan pendidikan berupa layanan *Learning Management System* (LMS) sebagai salah satu saranan pembelajaran yang banyak digunakan[2], salah satu layanan LMS yang sering digunakan adalah LMS moodle yang memanfaatkan teknologi informasi bersifat *open source* [3]. Kapasitas server merupakan faktor penting yang harus diperhatikan, karena dengan banyaknya pengguna yang mengakses layanan server terutama LMS dapat menyebabkan terjadinya *overload* atau kelebihan kapasitas yang dapat menyebabkan server menjadi *down* [4].

Load balancing dapat menjadi solusi untuk mengatasi permasalahan server down, dengan membagi beban kerja ke beberapa server secara seimbang [5]. *Load balancing* dapat diterapkan dengan berbagai algoritma salah satunya *round robin* dan *least*

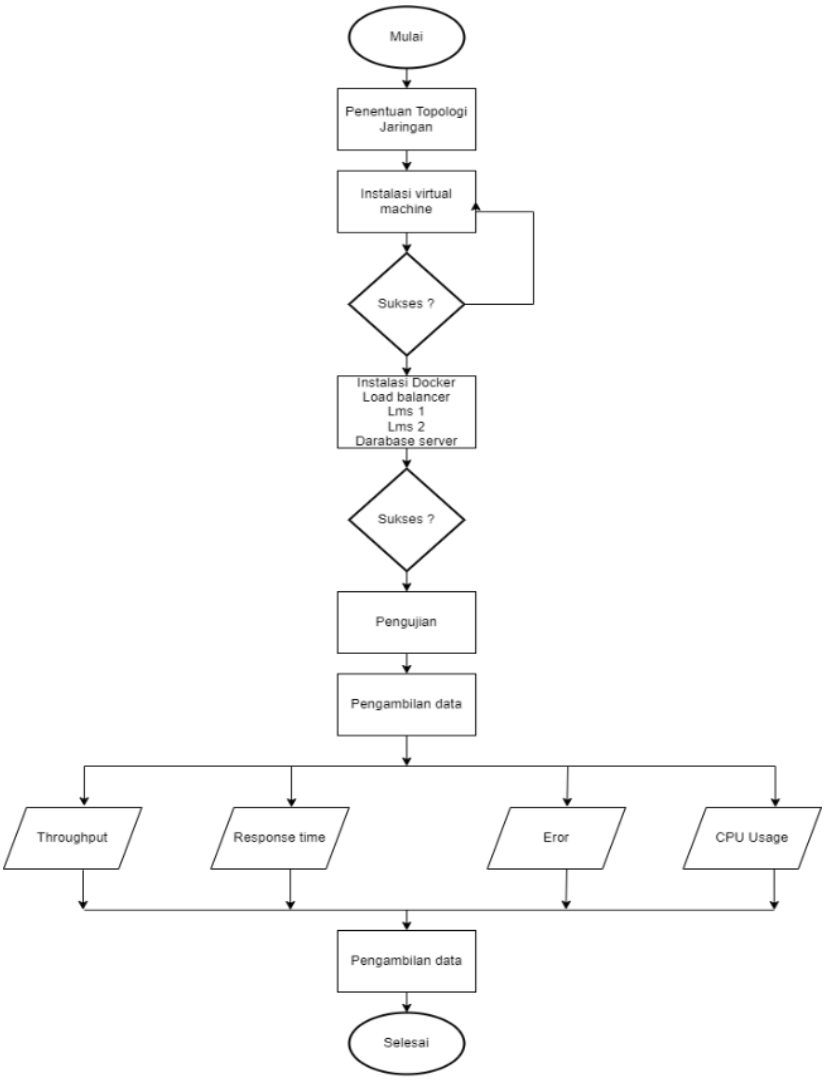
connection. *Round robin* adalah algoritma yang banyak digunakan dalam implementasi *load balancing*, algoritma ini akan membagi beban kerja secara seimbang tanpa mempertimbangkan parameter apapun [6][7]. Sedangkan *least connection* adalah algoritma yang akan memprioritaskan pembagian beban kerja ke server yang memiliki koneksi paling sedikit [8].

Penelitian dengan metode *load balancing* umumnya diterapkan secara langsung pada peragkat server, namun hal tersebut memerlukan konfigurasi yang kompleks karena program harus dibuat dari awal. Docker dapat menjadi jawaban, docker merupakan platform yang dapat membungkus aplikasi dalam sebuah wadah yang disebut *container* [9], docker melakukan proses *deployment* dengan aplikasi yang sudah berbentuk paket, sehinnga dapat mempersingkat proses *deployment* aplikasi [10]. Berdasarkan persoalan tersebut penulis mengambil judul penelitian "ANALISIS PERFORMANSI *LOAD BALANCING LMS MOODLE* PADA *DOCKER CONTAINER*"

2. METODOLOGI PENELITIAN

2.1. Alur penelitian

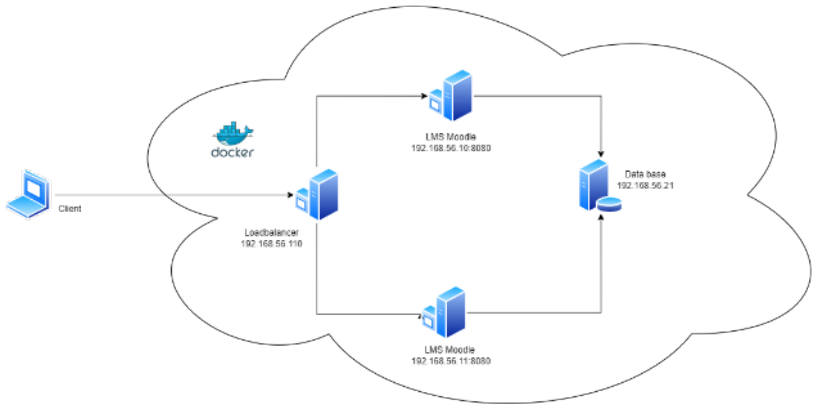
Tahap pertama pada penelitian ini adalah menentukan topologi jaringan yang akan digunakan sebagai acuan dari penerapan *load balancing*. Sistem *load balancing* akan diterapkan pada *docker container*. *Docker container* berperan sebagai wadah untuk menjalankan semua aplikasi yang akan digunakan. Konfigurasi *system load balancing* diantaranya adalah melakukan instalasi Nginx. Nginx merupakan web server *open source* dengan kemampuan tinggi yang dapat menangani masalah performa dari *web server* dengan koneksi aktif lebih dari 10.000 koneksi dalam waktu bersamaan dengan penggunaan memori yang lebih rendah dibandingan dengan web server lainnya. Selain itu Nginx juga dapat berperan sebagai *load balancer* atau pembagi beban kerja ke beberapa server yang berjalan. Selanjutnya melakukan instalasi Moodle sebagai platform *learning management system* yang akan digunakan serta melakukan instalasi dan konfigurasi database Mariadb sebagai tempat penyimpanan data untuk *learning management system*. Lalu mengatur konfigurasi Nginx sebagai *load balancer* dengan menggunakan dua algoritma yaitu *round robin* dan *least connection*. Kemudian melakukan pengujian menggunakan *stress tools* Httperf untuk mengambil parameter *throughput*, *response time* dan jumlah error. Selain itu penelitian ini juga akan menggunakan perintah "sar" untuk memantau penggunaan CPU saat dilakukan pengujian *system load balancing*.



Gambar 1. Alur Penelitian

2.2. Topologi jaringan

Topologi jaringan ditunjukan pada Gambar 2. Pada penelitian ini menenggunakan 4 server dan 1 client dengan 1 server sebagai server *load balancer*, 2 server sebagai server *learning management system*, 1 sever sebagai database server, dan 1 client bertugas sebagai penggujian server *load balancing*.



Gambar 2. Topologi Jaringan

2.3. Parameter pengujian

a. *Throughput*

Throughput adalah besarnya data yang diterima dalam kurun waktu tertentu, didalam sebuag konteks jaringan. Pengujian throughput digunakan untuk mengetahui seberapa cepat sebuah system dapat menangani request yang masuk pada jaringan dalam periode tertentu [11].

b. *Response time*

Response time adalah parameter yang mencakup kecepatan server dalam menerima permintaan dari pengguna atau waktu yang dibutuhkan oleh system untuk menanggapi response dari pengguna, waktu tersebut merupakan waktu saat proses pengeriman dan penerimaan paket data [12].

c. *Request error*

Request error adalah kondisi dimana server gagal melayani permintaan dari pengguna, sehingga pengguna tidak dapat mendapatkan request yang diminta ke system[13].

d. CPU *usage*

CPU *usage* atau penggunaan CPU adalah jumlah penggunaan sumber daya yang dibutuhkan saat menjalankan sebuah program atau system yang berjalan dalam proses komputasi [13].

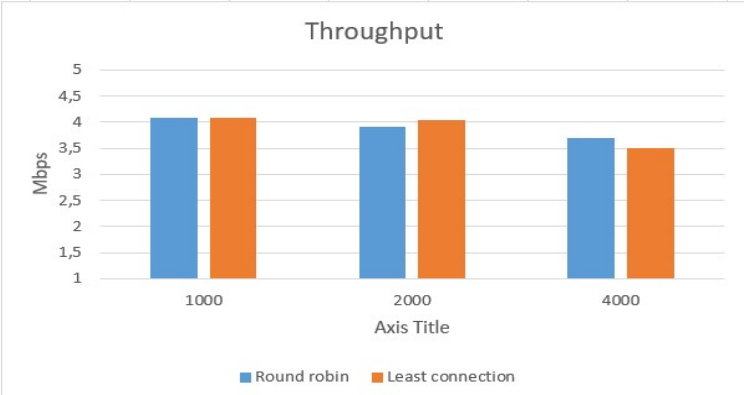
3. HASIL DAN PEMBAHASAN

3.1. *Throughput*

Pengukuran throughput dilakukan untuk mengetahui besar (bandwith) atau kecepatan jumlah informasi/data yang dapat dikirimkan dalam satuan waktu dan kondisi tertentu. Nilai rata rata throughput didapatkan dengan melihat NET I/O pada hasil pengujian melalui stress tools HTTPerf dalam satuan KB/s dan dikonversi menjadi satuan kbps.

Tabel 3.1 Hasil pengujian throughput

No	Beban		Throughput		Satuan
	Jumalah Request	Request Perdetik	Round Robin	Least Connection	
1	1000	100	4,07716	4,08333	Mbps
2	2000		3,9012	4,04308	
3	5000		3,69096	3,49116	



Gambar 3. Hasil pengujian *throughput*

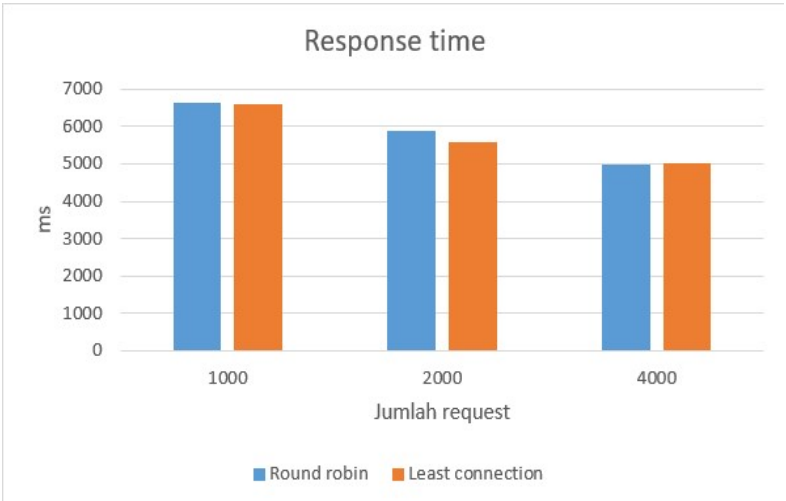
Pada pengukuran dari hasil perhitungan rata-rata tiap jumlah koneksi didapatkan hasil seperti pada table 4.2 terlihat penurunan nilai *throughput*, hal tersebut disebabkan oleh penambahan jumlah koneksi secara drastis sehingga membuat kinerja server lms menjadi menurun. Nilai *throughput* pada algoritma *least connection* mendapatkan hasil yang lebih baik dari algoritma *round robin*, hal ini dikarenakan algoritma *least connection* akan membagi *request* ke server yang memiliki beban terkecil sehingga proses akan lebih cepat dilakukan.

3.2. Analisis *Response Time*

Pengukuran nilai dari response time dilakukan untuk mengetahui kecepatan atau waktu yang dibutuhkan server untuk menanggapi pemintaan dari user.

Tabel 3.2 Hasil pengujian response time

No	Beban		Round Robin	Least Connection	Satuan
	Jumlah Request	Request per detik	Server LMS 1	Server LMS 2	
1	1000	100	6635,455	6616,645	ms
2	2000		5869,555	5569,23	
3	5000		4985,31	5032,535	



Gambar 4. Hasil pengujian response time

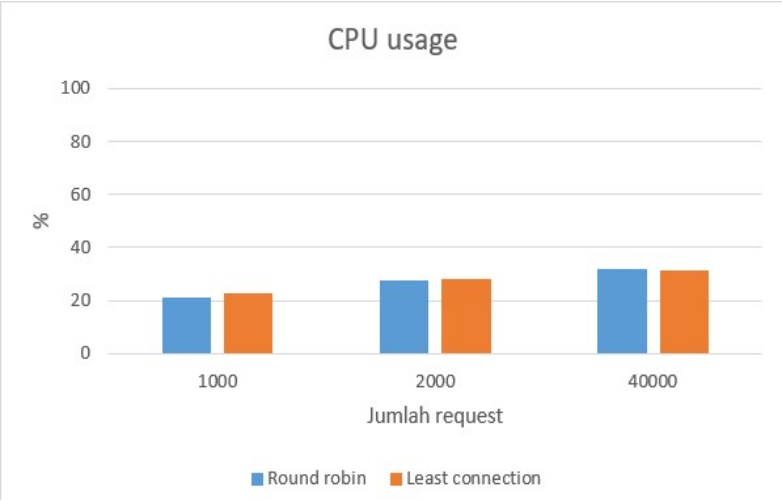
Hasil pengujian dari pengujian response time pada masing masing algoritma dengan jumlah request 1000, 2000, 5000, menunjukan hasil bahwa nilai *response time* semakin menurun, hal tersebut dikarenakan oleh meningkatnya jumlah eror dari setiap pengujian, hal itu menyebabkan nilai dari parameter *response time* sedikit mengalami penurunan, semakin banyak eror yang terjadi maka akan mempengaruhi hasil dari response time.

3.3. Analisis CPU *usage*

Pengukuran CPU *usage* dilakukan dengan tujuan untuk mengetahui informasi beban kerja yang ditanggung oleh server *load balancer* LMS Moodle pada penelitian ini. Nilai dari penggunaan CPU dilihat menggunakan *tools* HTTPerf dengan satuan persen. Nilai dari rata-rata penggunaan CPU masing masing server dan *request* disajikan pada Table 4.2.

Tabel 3.3 Hasil pengujian penggunaan cpu

No	Beban		Throughput	
	Jumlah Request	Request Per detik	Round Robin	Least Connection
1	1000	100	20,92	23
2	2000		27,57	27,98
3	5000		32,14	31,23



Gambar 5. Hasil pengujian penggunaan CPU

Hasil dari rata-rata penggunaan CPU pada algoritma *round robin* dan *least connection* mendapatkan hasil bahwa nilai dari penggunaan CPU berbanding lurus dengan bertambahnya jumlah *request* yang diterima oleh server selama berlangsungnya proses pengujian, semakin banyak jumlah *request* yang diterima server maka nilai CPU-nya akan meningkat. Berdasarkan nilai rata-rata penggunaan CPU pada Gambar 4.6. *system load balancing* masih dapat berjalan dan bekerja secara optimal hingga jumlah *request* ke 4000.

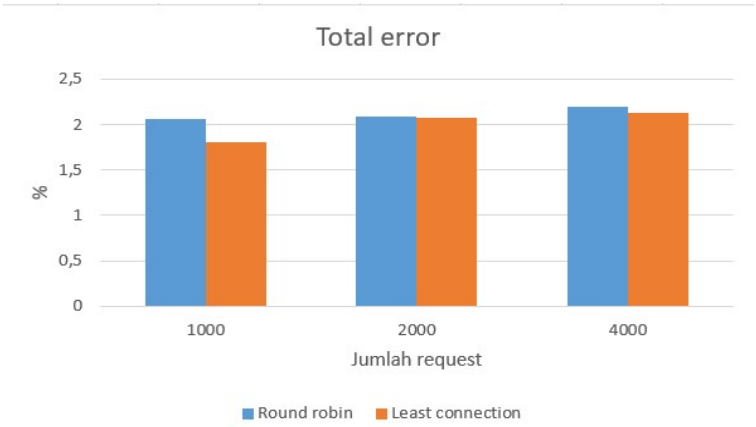
3.4. Analisis Eror

Nilai dari parameter error adalah jumlah kesalahan yang ditemui selama pengujian berlangsung. Dengan menggunakan HTTPerf dapat menentukan jumlah error yang terjadi dan juga dapat menentukan penyebab dari error yang terjadi saat pengujian berlangsung,

penyebab error yang dapat ditentukan antara lain : *client-timo, socket-timo, connrefused* dan *conreset.*

Tabel 3.4 Hasil pengujian jumlah error

No	Beban		Error	
	Jumalah *Request*	*Request* Per Detik	*Round Robin*	*Least connection*
1	1000	100	2,06	1,8
2	2000		2,0925	2,0725
3	5000		2,195	2,125



Gambar 6. Hasil pengujian jumlah error

Pada penelitian ini, semua eror yang terjadi saat pengujian berlangsung disebabkan oleh *connection reset* saat user melakukan *request. Connection reset* adalah kondisi dimana koneksi TCP gagal karena reset dari server, kondisi dimana paket tak terduga masuk kedalam server dan server akan merespon dengan mengembalikan paket tersebut kepada pengirim koneksi. Ada dua kemungkinan terjadinya hal tersebut, yaitu paket tersebut merupakan paket SYN awal yang mencoba membuat koneksi kepada port server dan yang kedua adalah karena client mengirim *request* saat server menutup akhir dari koneksinya. Selain itu penyebab terjadinya error juga disebabkan oleh jumlah *request* yang mengalami kenaikan secara drastis.

4. KESIMPULAN

Berdasarkan hasil pengujian dan proses analisis data didapatkan kesimpulan sebagai berikut :

a. Pada pengujian *Throughput*, algoritma *least connection* mendapatkan hasil yang lebih baik pada skema pengujian 1000 dan 2000 request sedangkan algoritma *round robin* mendapatkan hasil yang lebih baik pada pengujian dengan 4000 *request*.
b. Pada pemantauan rata-rata *response time* didapatkan bahwa terjadi penuruan nilai *response time* dari setiap pengujian, hal tersebut terjadi karena banyaknya *request* yang gagal pada saat pengujian berlangsung.
c. Presentase penggunaan CPU pada sever berbading lurus dengan meningkatnya jumlah *request* yang diberikan, yang berarti semakin banyak *request* yang dilayani server maka semakin tinggi presentase penggunaan CPU. Presentase CPU pada setiap variasi pengujian yaitu 20,92 %, 27,57% dan 32,14 % pada algoritma *round robin* dan 23%, 27,98% dan 31,23% pada algoritma *least connection*. Berdasarkan

nilai rata-rata penggunaan CPU tersebut, *system load balancing* masih dapat berjalan secara optimal sampai *request* ke 4000.

d. Hasil pengamatan jumlah error didapatkan data bahwa semakin tinggi jumlah *request* yang di berikan ke server maka tingkat terjadinya error semakin bertambah
e. Implementasi *system load balancing* untuk server *learning management system* Moodle menggunakan Nginx pada *docker container*, menunjukan hasil yang baik dan optimal sampai dengan *request* ke 4000.

DAFTAR PUSTAKA

[1] D. H. Jayani, "Penggunaan Internet di Kalangan Siswa Sekolah Semakin Meningkat," */databoks.katadata.co.id*, 2021. https://databoks.katadata.co.id/datapublish/2021/05/03/tren-siswa-sekolah-menggunakan-internet-semakin-meningkat (*accessed* Mar. 28, 2022).

[2] I. G. N. WIRAGUNAWA, "Pemanfaatan Learning Management System (LMS) dalam Pengelolaan Pembelajaran Daring pada Satuan Pendidikan," *J. Inov. Pendidik. Berbantuan Teknol. 82 Vol. 2 No. 1 Februari 2022, e-ISSN 2797-0140 | p-ISSN 2797-0590*, vol. 2, no. 1, pp. 82–89, 2022.

[3] K. Sara, F. L. Witi, and A. Mude, "Implementasi E-Learning Berbasis Moodle di Masa Pandemi Covid 19," *J. Adm. Educ. Manag.*, vol. 3, no. 2, pp. 181–189, 2020, doi: 10.31539/alignment.v3i2.1813.

[4] 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.

[5] 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: 10.26905/jtmi.v6i1.3792.

[6] H. Rodiawati and K. Komarudin, "Pengembangan E-Learning Melalui Modul Interaktif Berbasis Learning Content Development System," *J. Tatsqif*, vol. 16, no. 2, pp. 172–185, 2018, doi: 10.20414/jtq.v16i2.190.

[7] H. Triangga, I. Faisal, and I. Lubis, "Analisis Perbandingan Algoritma Static Round-Robin dengan Least-Connection terhadap Efisiensi Load Balancing pada Load Balancer Haproxy," *InfoTekJar (Jurnal Nas. Inform. dan Teknol. Jaringan)*, vol. 4, no. 1, pp. 70–75, 2019, doi: 10.30743/infotekjar.v4i1.1688.

[8] M. Sholeh, W. Yahya, and P. H. Trisnawan, "Implementasi Load Balancing menggunakan Algoritme Least Connection dengan Agen Psutils pada Web Server", vol. 3, no. 1. repository.ub.ac.id, 2019.

[9] C. Mukmin and E. S. Negara, "Analisis Kinerja Redistribusi Routing Protokol Dinamik (Studi Kasus : RIP, EIGRP, IS-IS)," *Klik - Kumpulan Jurnal Ilmu Komputer*, vol. 6, no. 3. scholar.archive.org, p. 284, 2019, doi: 10.20527/klik.v6i3.262.

[10] S. Dwiyatno, E. Rakhmat, and O. Gustiawan, "Implementasi Virtualisasi Server Berbasis Docker Container," *Prosisko*, vol. 7, no. 2, pp. 165–175, 2020, [Online]. Available: https://e-jurnal.lppmunsera.org/index.php/PROSISKO/article/view/2520/.

[11] R. Arjeko, "Perbandingan dan Analisis Throughout Load Balance NTH dan PCC untuk Optimalisasi Trafik Mangle," *Jurnal Sains Dan Komputer (Infact)*. journal.ukrim.ac.id, 2021, [Online]. Available:

http://journal.ukrim.ac.id/index.php/JIF/article/download/303/240.

[12] K. Ekmawan, "Analisa Implementasi Load Balancing Round Robin Dan Least Connection Pada Web Server (Studi Kasus PT UCC)," *Jtika*, vol. 3, no. 2, pp. 244–252, 2021, [Online]. Available: http://jtika.if.unram.ac.id/index.php/JTIKA/article/view/158.

[13] M. S. Pradana and A. Prapanca, "Analisis Performa Load Balancing Algoritma Weighted Round Robin di Infrastruktur BPBD Provinsi Jawa Timur," *J. Informatics Comput. Sci.*, vol. 1, no. 02, pp. 109–114, 2020, doi: 10.26740/jinacs.v1n02.p109-114.