InComTech: Jurnal Telekomunikasi dan Komputer
vol.11, no.3, Desember 2021, 210-220
http://publikasi.mercubuana.ac.id/index.php/Incomtech
P-ISSN: 2085-4811 E-ISSN: 2579-6089

Analisis Kinerja *Load Balancing* pada Server Web Menggunakan Algoritma *Weighted Round Robin* pada Proxmox VE

Bongga Arifwidodo*, Vassa Metayasha, Syariful Ikhwan

Program Studi Teknik Telekomunikasi, Institut Teknologi Telkom Purwokerto,
Jl. D.I Panjaitan No. 128 Purwokerto, Indonesia
*Email Penulis Koresponden: bongga@ittelkom-pwt.ac.id

Abstrak:
Peningkatan jumlah lalu lintas server web mengakibatkan peningkatan kinerja server. Kondisi penggunaan satu server dirasa kurang efektif, karena dapat menyebabkan server down jika mendapat beban trafik yang berlebihan. Penelitian ini menggunakan algoritma *Weighted Round Robin* (WWR) karena mampu mempertimbangkan beban server berdasarkan kapasitas sumber daya server. Kemudian diimplementasikan pada lingkungan virtualisasi Proxmox VE. Pengukuran kinerja jaringan dilakukan dengan mengirimkan trafik *request* HTTP ke server web. Untuk mendapatkan pembagian beban kinerja yang optimal dari skenario *Weighted Round Robin load sharing*, WRR 2:1, WRR 3:1, WRR 4:1, dan WRR 5:1 perlu dilakukan penelitian pengukuran *Quality of Service* dengan standar TIPHON serta penggunaan CPU. Hasil penelitian ini, pembagian kerja skenario penjadwalan yang paling baik adalah pada WRR 2:1, karena mampu membagi beban secara merata antara kedua server. Sehingga menghasilkan rata-rata *throughput* tertinggi 6.717 Mbit/s, nilai *delay* terendah 0.505 ms, dan tidak ada paket yang hilang. WRR 2:1 juga membuat kedua server web tersebut mendapatkan beban kerja yang seimbang, dengan perbedaan penggunaan CPU pada koneksi 5000 mencapai 2,244%, koneksi 10000 mencapai 4,528%, dan koneksi 15000 mencapai 3,111%.

This is an open access article under the CC BY-NC license

***Kata Kunci**:*
Weighted round robin;
Virtualization;
Proxmox;

Riwayat Artikel:
Diserahkan 4 Mei 2021
Direvisi 26 Oktober 2021
Diterima 13 November 2021
Dipublikasi 31 Desember 2021

DOI:
10.22441/incomtech.v11i3.11775

1. PENDAHULUAN

Kinerja server web dan *database* sebagai penyedia konten media selalu diharapkan dapat memenuhi semua kebutuhan pengguna [1]. Masalahnya dimulai ketika satu server menerima permintaan dari dari penggunaan dengan jumlah yang besar, menyebabkan server digunakan secara berlebihan. Beban ekstra yang terjadi

pada server akan mengakibatkan server *down* karena tidak dapat lagi menerima jumlah *request* dari pengguna. Oleh karena itu diperlukan suatu sistem yang dapat meminimalkan *downtime* server [2]. Penyeimbangan beban adalah mekanisme mengalokasikan dan mengalokasikan kembali beban di antara sumber daya yang tersedia untuk meningkatkan kinerja dan memaksimalkan *throughput* pada waktu respons minimum dan biaya minimum [3]. Saat menerapkan teknik *load balancing*, diperlukan penyeimbang beban dengan menggunakan algoritma penjadwalan. Sehingga algoritma *load balancer* dapat meneruskan *request packet* dari pengguna ke server [4]. Salah satu algoritma penjadwalan *load balancing* adalah *weighted round robin*. Algoritma penjadwalan *round-robin* berbobot secara manual memasukkan bobot atau parameter untuk setiap node *cluster* berdasarkan sumber daya, sehingga penjadwal pekerjaan akan memprioritaskan pekerjaan untuk server itu di atas server lain. Berbeda dengan algoritma *least-connected* dan IP *hash*, dimana melakukan prioritas pembagian dari beban kinerja yang paling rendah dan berdasarkan *request* dari pengguna (menggunakan alamat IP dari pengguna). Sehingga server akan selalu menerima *request* dari alamat IP yang berbeda [5-7].

Keunggulan menarik dari teknik *load balancing* algoritma WRR umumnya diterapkan pada server web fisik yang memiliki spesifikasi yang berbeda [8] bahwa kinerja server web lebih optimal dan stabil. Namun algoritma ini bisa diterapkan pada layanan komputasi awan pada Proxmox dan Xenserver. Seperti yang diungkapkan [9] bahwa kedua *hypervisor* tipe 1 tersebut mampu menjalankan sistem virtualisasi dengan baik. Tantangan selanjutnya adalah *load balancing* menjadi kurang efisien ketika diimplementasikan menggunakan server fisik, karena kebutuhan untuk mengubah atau menambah perangkat jika server tidak dapat mengakomodasi permintaan yang semakin banyak. Masalah tersebut dapat diatasi dengan membangun sistem virtualisasi yang dapat menampung jumlah komputer server dalam satu komputer server fisik dan dapat menjalankan sistem operasi yang berbeda secara bersamaan [10]. Virtualisasi adalah proses mengadopsi *instance* secara virtual dari mesin keadaan fisik untuk mencapai pemanfaatan penuh server [11].

Penggunaan server web tidak semuanya yang digunakan memiliki spesifikasi yang sama, dari latar belakang di atas penulis melakukan penelitian menggunakan teknik *load balancing* yang diterapkan pada sistem virtualisasi Proxmox dan menggunakan dua server yang memiliki spesifikasi yang berbeda. Algoritma *load balancing* yang digunakan menggunakan algoritma *Weighted Round Robin* (WWR) karena mampu mengatur jumlah bobot pada setiap server walaupun server memiliki spesifikasi berbeda [11]. Sehingga didapatkan nilai parameter QoS (*Quality of Service*) yang berstandar TIPHON, *throughput*, *delay*, *packet loss*, dan penggunaan CPU pada kinerja *load balancing* yang diterapkan pada server virtualisasi Proxmox.

2. METODE

2.1 Alur Penelitian

Tahapan pertama menentukan topologi jaringan yang akan digunakan sebagai dasar dari arsitektur sistem jaringan *load balancing*. Sistem *load balancing* pada penelitian ini diimplementasikan dalam teknologi virtualisasi yaitu Proxmox VE

6.1. Proxmox VE berperan sebagai *hypervisor type* 1 yang digunakan sebagai tempat atau wadah untuk membangun server virtual [12]. Selanjutnya membuat mesin virtual untuk setiap konfigurasi server pada setiap server sesuai fungsinya. Beberapa konfigurasi server *load balancing* diantaranya melakukan instalasi HAProxy. HAProxy menawarkan layanan *load balancing* ke layanan berbasis HTTP dan TCP, seperti layanan yang terhubung ke internet dan aplikasi berbasis web. Bergantung pada algoritma penjadwalan sebagai penyeimbang beban yang dipilih, sehingga HAProxy dapat melayani ribuan koneksi yang terjadi [13]. Selanjutnya melakukan konfigurasi Apache2 pada server web dengan menambahkan paket perangkat lunak pembelajaran *Moodle* sebagai tampilan layanan server web. Pada saat melayani banyak pengguna secara bersamaan dan memproses lebih banyak SQL dalam satu waktu diperlukan konfigurasi server *database* dengan sistem manajemen *database* relasional MariaDB [14] untuk penyimpanan data. Konfigurasi *load balancing* menggunakan algoritma penjadwalan *weighted round robin*, karena kemampuannya menangani server dengan ketersediaan sumber daya perangkat yang berbeda-beda [15]. Selanjutnya melakukan pengujian sistem *load balancing* pada sistem yang berhasil dibangun.

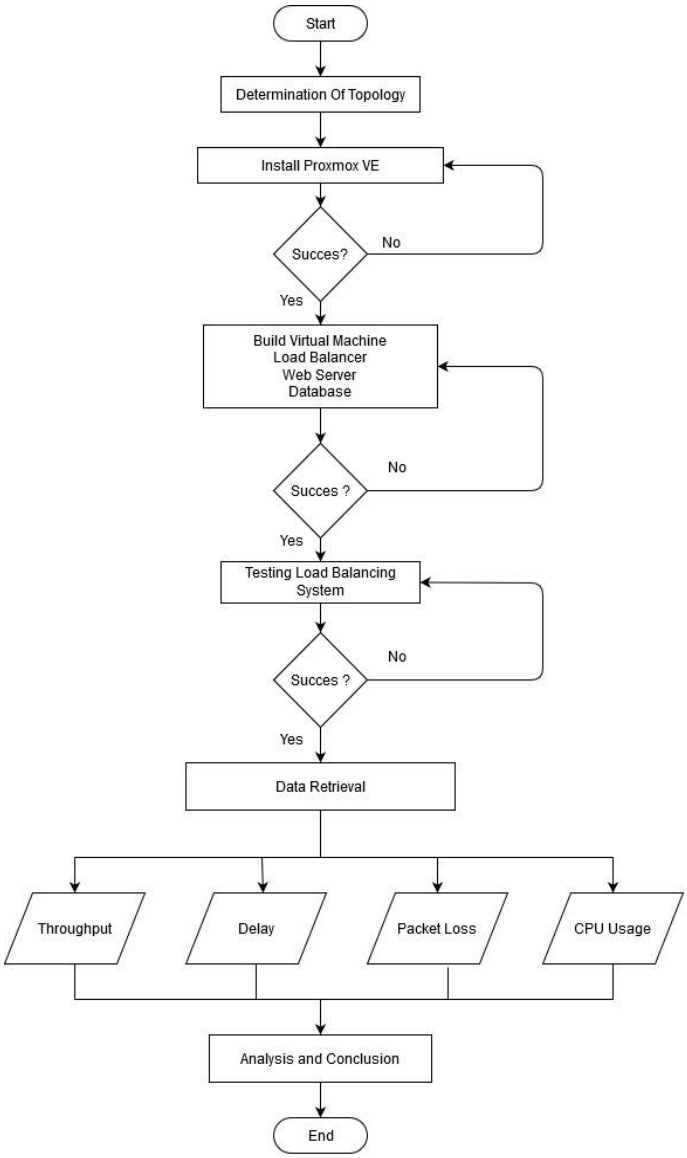
Apache Benchmark digunakan sebagai *stress tool* untuk memuat lalu lintas ke sistem penyeimbangan beban. Kami menggunakan aplikasi Wireshark untuk menangkap paket data selama proses transmisi data. Pengukuran kualitas jaringan dilakukan dengan mengirimkan trafik *request HTTP* ke HAProxy dalam beberapa skenario *Weighted Round Robin load sharing*, yaitu WRR 2:1, WRR 3:1, WRR 4:1, dan WRR 5:1 (Tabel 1).

Tabel 1. Skenario WRR

Pembagi Beban	Server Web I	Server Web II
WRR 2:1	2	1
WRR 3:1	3	1
WRR 4:1	4	1
WRR 5:1	5	1

Algoritma WRR 2:1 artinya *load balancer* akan memberikan beban dua *request* ke server web I dan 1 *request* ke server web II, selanjutnya skenario pembagian beban 3:1 artinya *load balancer* akan memberikan beban 3 *request* ke server web I dan 1 *request* ke server web II, skenario pembagian beban 4:1 artinya load balancer akan memberikan beban 4 *request* ke server web I dan 1 kali *request* ke server web II, dan skenario pembagian beban 5:1 artinya load balancer memberikan beban 5 kali *request* ke server web I dan 1 kali *request* ke server web II.

Proses analisis dilakukan dengan mengukur nilai parameter *Quality of Service* (QoS) dan penggunaan CPU. Tahapan akhir adalah analisis data yang telah berhasil diperoleh dari pengujian untuk mengetahui performansi load balancing menggunakan algoritma *Weighted Round Robin* yang dijalankan pada lingkungan virtualisasi *hypervisor* tipe 1 yaitu Proxmox VE. Kesimpulan penelitian diambil dengan memperhatikan tujuan dari penelitian agar memperoleh hasil yang sesuai. Saran diberikan untuk mendorong pada penelitian yang akan mengambil tema yang serupa. Secara keseluruhan alur penelitian ini dapat dilihat pada Gambar 1.



Gambar 1. Alur Penelitian

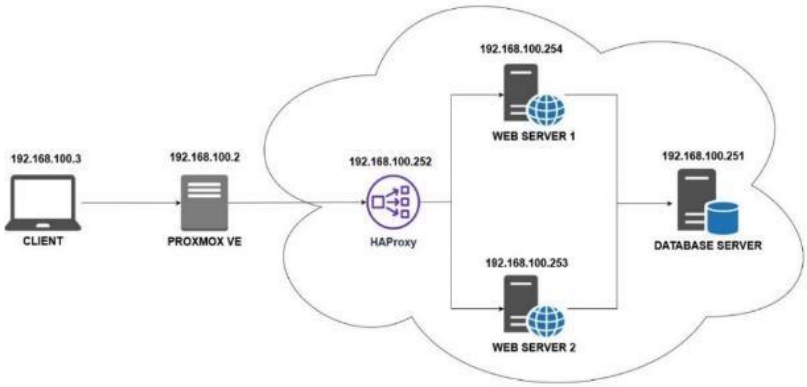
2.2. Skenario Penelitian

Topologi jaringan fisik ditunjukkan pada Gambar 2.(a) dan topologi logik (pengujian) ditunjukkan pada Gambar 1.(b). Kedua topologi mempunyai hal yang sama yaitu terdiri dari sebuah klien yang berperan untuk mengirim dan menerima data dari atau ke server. Sebuah penyeimbang beban server untuk membagi beban lalu lintas ke server web, sejumlah 2 buah server web untuk menampilkan layanan *Moodle*, dan sebuah server basis data untuk penyimpanan data dari server web. Perbedaannya, topologi jaringan perlu mengakses sebuah server untuk membangun lingkungan virtualisasi, sedangkan dalam topologi pengujian pengguna langsung mengakses alamat IP penyeimbang beban.

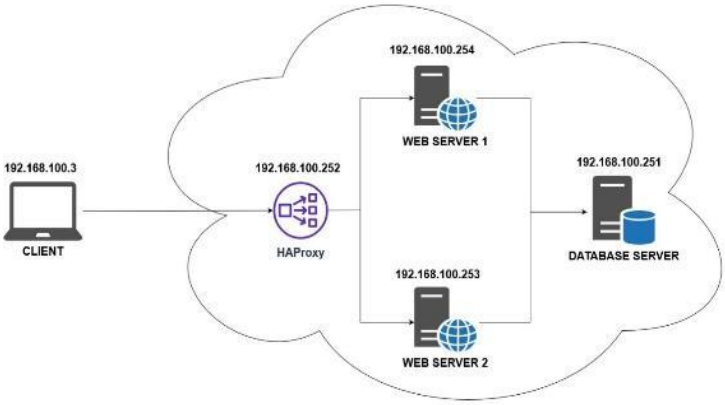
Pengujian bertujuan untuk mengetahui QoS pada layanan sistem *load balancing* masing-masing algoritma. Peneliti akan menguji 3 skenario sesuai Tabel 1 dan dilanjutkan dengan uji Qos sesuai Tabel 2.

Tabel 2. Uji QoS

	Jumlah koneksi	*Concurrency*	Jumlah Pengujian
Uji ke 1	5000	10	30
Uji ke 2	10000	10	30
Uji ke 3	15000	10	30



(a)



(b)

Gambar 2. Topologi (a) topologi fisik (b) topologi logik

2.3. Konfigurasi Penelitian

Pada tahap ini kami melakukan konfigurasi sistem. Implementasi Proxmox VE versi 6.1 sebagai lingkungan server virtual, dengan mengkonfigurasi server web dan penambahan instalasi perangkat lunak Apache2. Server *database* kami mengkonfigurasikan MariaDB kemudian menghubungkannya ke server web. Konfigurasi sistem *load balancing* menggunakan *software* HAProxy yang dibangun pada sistem operasi Ubuntu server 18.04. Algoritma WRR mampu menyeimbangkan kedua server yang memiliki spesifikasi berbeda sehingga server web yang digunakan dalam penelitian ini menggunakan 2 buah mesin virtual dengan spesifikasi seperti pada Tabel 3 berikut.

Tabel 3. Data Spesifikasi

	Server Web I	Server Web II
OS	Ubuntu server 18.04	Ubuntu server 18.04
CPU	2 core	1 core
RAM	4 GB	2 GB

2.4. Parameter Penelitian

Quality of Service (QoS) didefinisikan sebagai kemampuan untuk menjamin kebutuhan jaringan tertentu seperti *bandwidth*, *latency*, *jitter*, dan kehandalan untuk memenuhi *Service Level Agreement* (SLA) antara penyedia aplikasi dan pengguna akhir (*end user*) [16]. *Quality of Service* (QoS) sangat ditentukan oleh kualitas dari sebuah jaringan yang digunakan. Adanya beberapa faktor yang dapat mempengaruhi kualitas dari nilai QoS, seperti redaman, distorsi dan *noise* [17].

a. Throughput

Throughput adalah nilai kecepatan dari transfer data efektif yang diukur dalam satuan bps. *Throughput* merupakan jumlah total kedatangan paket yang berhasil diterima dengan interval waktu tertentu dan dibagi oleh durasi interval waktu tersebut [18]. Tabel 4 menunjukan nilai standar TIPHON untuk *throughput*.

$$Throughput = \frac{Jumlah\ data\ yang\ dikirim}{Waktu\ pengiriman\ data} \quad (1)$$

Tabel 4. *Throughput* [19]

Kategori	*Throughput*	Indeks
sangat memuaskan	>2,1 Mps	4
memuaskan	1200 kbps - 2,1 Mbps	3
kurang memuaskan	700 - 1200 kbps	2
tidak memuaskan	< 700 kbps	1

b. *Delay*

Delay adalah total waktu yang dilalui oleh suatu paket dari pengirim ke penerima pada sebuah jaringan. *Delay* antar user pengirim ke penerima pada dasarnya tersusun atas *hardware latency*, *delay* transmisi dan *delay* akses [17]. Tabel 5 menunjukan nilai standar TIPHON untuk *delay*.

$$Delay\ (\text{ms}) = \frac{Total\ Delay}{Total\ paket\ yang\ diterima} \quad (2)$$

Tabel 5. *Delay* [19]

Kategori	*Throughput*	Indeks
Sangat memuaskan	< 150 ms	4
Memuaskan	150 s/d 300 ms	3
Kurang memuaskan	300 ms s/d 450 ms	2
Tidak memuaskan	>450 ms	1

c. *Packet Loss*

Packet loss adalah parameter yang menggambarkan sebuah kondisi yang menunjukkan total paket yang hilang pada transfer data yang terjadi. *Packet loss* terjadi karena *collision* dan *congestion* pada jaringan [18]. Tabel 6 menunjukan nilai standar TIPHON untuk *packet loss*.

$$Packet\ Lo(\%) = \frac{paket\ data\ yang\ dikirim - paket\ data\ yang\ diterima}{paket\ data\ yang\ dikirim} \text{ x } 100\% \quad (3)$$

Tabel 6. *Packet Loss* [19]

Kategori	Besar *Packet Loss*	Indeks
Sangat bagus	0	4
Bagus	3	3
Sedang	15	2
Jelek	25	1

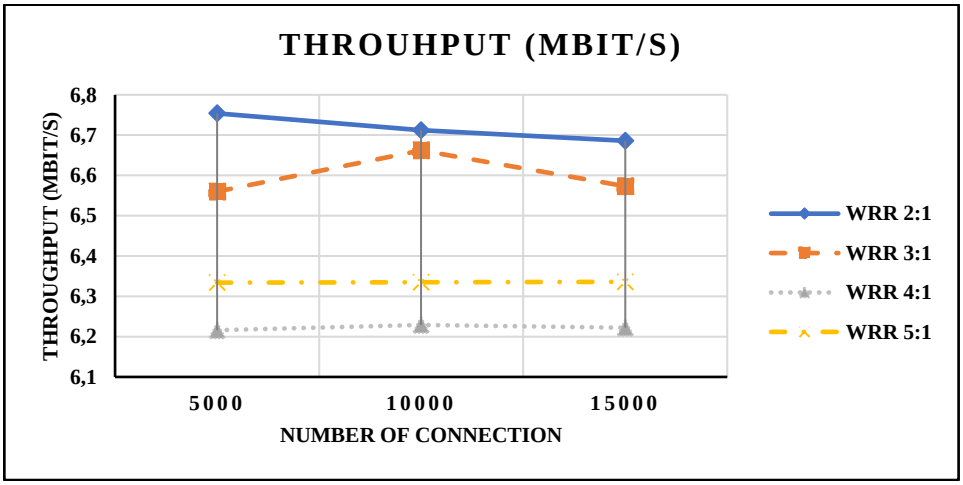
3. HASIL DAN PEMBAHASAN

Penelitian ini melakukan analisis kinerja *load balancing* menggunakan algoritma *Weighted Round Robin* pada Prosmox VE. Pengukuran unjuk kerja dilakukan dengan menganalisa mengirim trafik HTTP *request* pada beberapa scenario. Pada bagian ini hasil nilai dari 3 skenario pengujian masing-masing parameter QoS akan diuraikan. Pengelompokan standarisasi QoS (*throughput*, *delay* dan *packet loss*) berdasarkan standar TIPHON.

3.1 Pengukuran *Throughput*

Pengukuran parameter *throughput* dilakukan untuk mengetahui kecepatan transfer data aktual pada jaringan yang dibangun. Nilai parameter *throughput* diperoleh dengan membagi jumlah paket yang diterima dengan total waktu pengiriman paket.

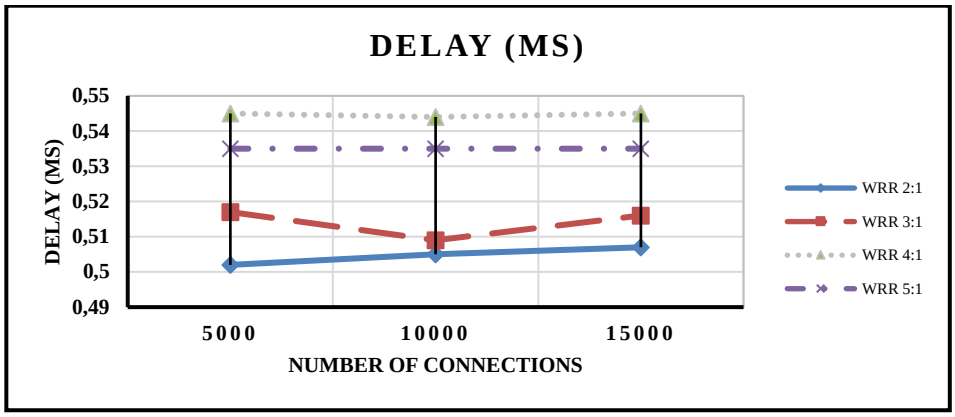
Berdasarkan Gambar 3, nilai rerata *throughput* sistem *load balancing* menggunakan algoritma *Weighted Round Robin* 2:1 mendapatkan rata-rata nilai *throughput* tertinggi dibandingkan skema pembagian bobot lainnya. Hal ini menandakan bahwa algoritma WRR dengan bobot 2:1 mampu menyeimbangkan kinerja kedua server web, sehingga *request* yang diberikan oleh user dapat dilayani dengan lebih maksimal.



Gambar 3. Hasil *Throughput*

3.2 Pengukuran *Delay*

Pengukuran *delay* dilakukan untuk mengetahui waktu yang dibutuhkan sebuah paket data yang dikirimkan dari user hingga paket data yang diterima oleh server selama proses transmisi. Gambar 4 memberikan informasi bahwa rerata nilai *delay* pada setiap skenario pengujian tidak menunjukan perbedaan nilai yang signifikan.



Gambar 4. Hasil *Delay* (ms)

Pada jumlah koneksi 5000, didapatkan rata-rata nilai *delay* untuk algoritma WRR 2:1 sebesar 0,502 ms, WRR 3:1 sebesar 0,517 ms, WRR 4:1 sebesar 0,545 ms, dan WRR 5:1 sebesar 0,535 ms. Pada algoritma WRR 2:1 mengalami kenaikan seiring dengan bertambahnya jumlah koneksi. Pada jumlah koneksi 10000 menjadi 0,505 ms dan pada jumlah koneksi 15000 mengalami kenaikan kembali menjadi 0,507 ms. Pada algoritma WRR 3:1 saat jumlah koneksi 10000 mengalami penurunan *delay* menjadi 0,509 ms dan saat jumlah koneksi 15000 mengalami kenaikan kembali menjadi 0,516 ms. Sedangkan pada algoritma WRR 4:1 memiliki nilai yang stabil (konsisten), dimana pada jumlah koneksi 10000 mendapatkan nilai *delay* 0,544 ms dan pada jumlah koneksi 15000 mendapatkan nilai *delay* 0,545 ms. Pada algoritma WRR 5:1 memiliki hasil yang stabil di seluruh jumlah koneksi, dimana pada jumlah koneksi 10000 dan 15000 memiliki nilai *delay* yang sama dengan jumlah koneksi 5000 sebesar 0,535 ms. Akan tetapi, untuk seluruh pengujian mendapatkan hasil *delay* dengan kategori "Sangat Memuaskan" yang menyesuaikan dengan standarisasi TIPHON. Hal ini ditunjukkan dengan nilai *delay* yang dihasilkan kurang dari 150 ms (<150 ms).

Hasil rata – rata nilai *delay* ini juga berkaitan dengan rata-rata nilai *throughput*, dimana semakin besar nilai *throughput* yang dihasilkan maka nilai *delay* yang dihasilkan akan semakin kecil. Berdasarkan data rata-rata nilai *delay* yang didapatkan, sistem *load balancing* dengan algoritma *Weighted Round Robin* dengan pembagian bobot server 2:1 mendapatkan rata-rata nilai *delay* yang lebih kecil sebesar 0,505 ms. Hal ini terjadi karena algoritma WRR 2:1 mendapatkan nilai *throughput* tertinggi, sehingga menghasilkan rata-rata nilai *delay* yang lebih kecil.

3.3 Pengukuran *Packet Loss*

Pengukuran parameter *packet loss* dilakukan untuk mengetahui berapa banyak paket yang hilang selama proses transmisi terjadi. Jika terjadi maka kualitas dari jaringan maupun sistem yang dibangun kurang bagus. Pada penelitian ini, nilai *packet loss* dilihat pada aplikasi *benchmarking* yang digunakan yaitu *Apache Benchmark*. Nilai dari *packet loss* dapat dilihat pada bagian "*Failed Request*". Hasil pengukuran packet loss dapat diihat pada Tabel 7.

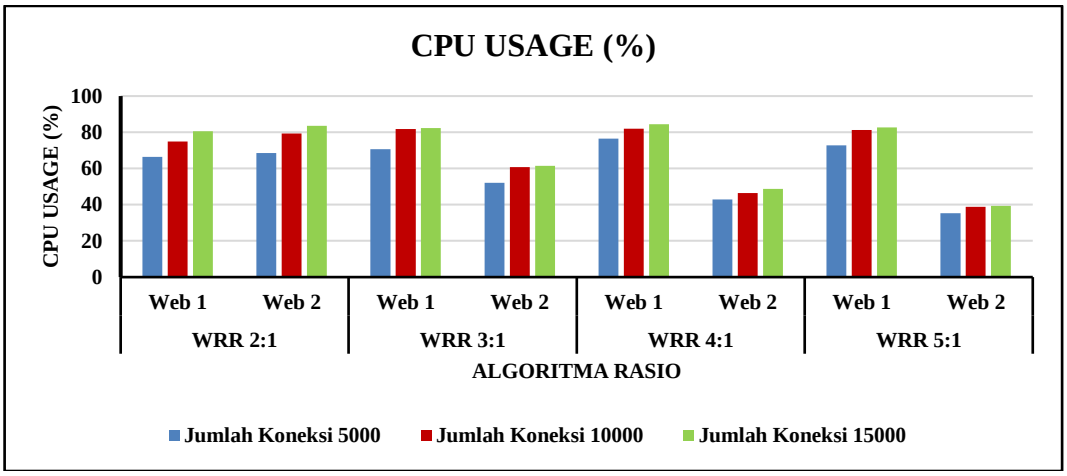
Tabel 7. Hasil *Packet Loss*

Number of Connections	***Packet loss* (%)**			
	WRR 2:1	**WRR 3:1**	**WRR 4:1**	**WRR 5:1**
5000	0.000	0.000	0.000	0.000
10000	0.000	0.000	0.000	0.000
15000	0.000	0.000	0.000	0.000

Nilai kelayakan parameter *packet loss* untuk seluruh skenario pengujian menunjukkan kategori "Sangat Memuaskan" yang menyesuaikan dengan standarisasi TIPHON. Hal ini ditunjukkan dengan paket data yang didapatkan bahwa tidak ada paket yang hilang pada pengujian diseluruh skenario. Hal ini bisa terjadi karena nilai *throughput* yang dihasilkan berkisar 6 Mbit/s hingga 6.7 Mbit/s. Besarnya *throughput* yang dihasilkan dapat meminimalisir terjadinya paket yang hilang pada saat proses transmisi data. Sehingga paket data yang dikirim dapat diterima seluruhnya dalam waktu singkat tanpa adanya paket yang hilang.

3.4 Pengukuran CPU *Usage*

Pengukuran CPU *usage* dilakukan untuk mengetahui penggunaan CPU pada masing-masing server web. Hal ini bertujuan untuk mengetahui apakah kedua server web mendapatkan beban yang seimbang atau hanya memberatkan ke salah satu server web saja. Merujuk pada pengertian *load balancing*, dimana sistem *load balancer* akan membagi beban secara merata dan seimbang kepada setiap server. Hasil rerata penggunaan CPU dapat dilihat pada Gambar 6.



Gambar 6. Hasil Penggunaan CPU

Dengan bertambahnya jumlah koneksi, maka akan bertambah pula proses kinerja CPU untuk menangani koneksi yang terjadi. Pada kondisi koneksi sejumlah 15000, server masih dapat menampung jumlah koneksi akan tetapi, server harus bekerja maksimal dengan penggunaan CPU 80% hingga 90%. Menariknya jika kondisi tersebut dipaksakan untuk melayani koneksi dalam jumlah banyak dan dalam waktu yang lama, maka server Proxmox akan memberikan peringatan tentang penggunaan CPU yang terlalu berlebih pada mesin server web virtual.

Berdasarkan uraian tersebut, server masih dapat bekerja optimal saat menerima jumlah koneksi 5000 hingga 10000 koneksi.

Semakin tinggi perbandingan rasio yang diberikan, maka selisih penggunaan CPU pada kedua server web akan semakin besar. Oleh karena itu, perlu mengetahui perbedaan spesifikasi pada setiap server terlebih dahulu agar pemberian bobot pada server lebih tepat dan performa Qos lebih baik. Pembagian yang tepat dengan spesifikasi server yang digunakan pada penelitian ini adalah rasio WRR 2:1. Keseimbangan antara kedua server web pada sistem *load balancing* dapat mempengaruhi performa sistem tersebut. Semakin seimbang penggunaan pada setiap server, maka sistem *load balancing* tersebut semakin baik, karena tidak memberatkan hanya ke satu server saja melainkan keseluruh server web mendapatkan beban yang sesuai walaupun memiliki spesifikasi yang berbeda.

4. KESIMPULAN

Hasil implementasi server web menggunakan algoritma penjadwalan WRR pada virtualisasi Proxmox menunjukan bahwa pembagian beban server dengan skenario WRR 2:1 lebih optimal (CPU *usage*) dalam pembagian beban kerja dengan menggunakan spesifikasi server yang berbeda. Hal ini ditunjukan hasil nilai parameter QOS sesuai standar TIPHON termasuk dalam kondisi sangat baik. Nilai *throughput* tertinggi sebesar 6.717 Mbit/s serta *delay* terendah 0,505 ms dan tidak ada *packet loss*.

REFERENSI

[1] 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, pp. 85-92, 2019, doi: 10.30595/jrst.v3i2.5165.

[2] R. Dani and F. Suryawan, "Perancangan dan Pengujian Load balancing dan Failover Menggunakan NginX," *Jurnal Ilmu Komputer dan Informatika*, vol. 3, no. 1, pp. 43-50, 2017, doi: 10.23917/khif.v3i1.2939.

[3] P. Singh, P. Baaga and S. Gupta., "Assorted load balancing algorithms in cloud computing: A survey," *Int. J. Comput. Appl*, vol. 143, no. 7, pp. 34–40, 2016, doi: 10.5120/ijca2016910258.

[4] S. Goyal, and M. K. Verma., "*Load balancing* techniques in cloud computing environment—A review," *Int. J. Adv. Res. Comput. Sci*, vol. 6, no. 4, p. 583–588, 2016.

[5] H.G. Tani and C.E, Amrani, " Smarter Round Robin Scheduling Algorithm for Cloud Computing and Big Data," *Journal of Data Mining and Digital Humanities*, 2018, doi: 10.46298/jdmdh.3104.

[6] D. Biswas and M. Samsuddoha, "Determining Proficient Time Quantum to Improve the," *Int. J. Mod. Educ. Comput. Sci*, vol. 11, no. 10, pp. 33-40, 2019, doi: 10.5815/ijmecs.2019.10.04.

[7] Y. Arta, "Penerapan Metode Round Robin Pada Jaringan Multihoming Di Computer *Cluster*," *Inf. Technol. J. Res. Dev*, vol. 1, no. 2, p. 26–35, 2017, doi: 10.25299/itjrd.2017.vol1(2).677.

[8] 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.2, pp. 109–114, 2019.

[9] S. Surahmat and A. Tenggono, "Analisis Perbandingan Kinerja Layanan Infrastructure As A Service Cloud Computing Pada Proxmox dan Xenserver", *MATRIK: Jurnal Manajemen, Teknik Informatika dan Rekayasa Komputer*, vol. 19, no. 1, pp. 9-16, Sep. 2019, doi: 10.30812/matrik.v19i1.434.

[10] S. R. Siregar, "Efisiensi Fisik Komputer Server dengan Menerapkan Proxmox Virtual Environment," *Journal of Computer System and Informatics (JoSYC)*, vol. 1, no. 2, pp. 83-86, 2020.

[11] Ashalatha R., J. Agarkhed and S. Patil, "Network virtualization system for security in cloud computing," *2017 11th International Conference on Intelligent Systems and Control (ISCO)*, 2017, pp. 346-350, doi: 10.1109/ISCO.2017.7856014.

[12] B. Harijanto and Y. Ariyanto, "Desain Dan Analisis Kinerja Virtualisasi Server Menggunakan Proxmox Virtual Environment," *Jurnal Simantec*, vol. 5, no. 1, pp. 17-24, 2015, doi: 10.21107/simantec.v5i1.1010.

[13] S. Levine and S. Wadeley, *Red Hat Enterprise Linux 7 Load Balancer Administration*, 2018. [Online]. Available: https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/7/html/load_balancer_administration/index.

[14] S. Widiono, "Experiments and Descriptive Analysis in The MariaDB Database Cluster System to Prepare Data Availability", *International Journal of Engineering Technology and Natural Sciences*, vol. 1, no. 1, pp. 42-48, Jul. 2019, doi: 10.46923/ijets.v1i1.24.

[15] 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, Jan. 2021, doi: 10.25299/itjrd.2021.vol5(2).5795.

[16] Openstack, *Quality of Service (QoS)*, 23 Agustus 2019. [Online]. Available: https://docs.openstack.org/ocata/id/networking-guide/config-qos.html. [Accessed 03 05 2021].

[17] R. Rasudin, "Quality of Services (Qos) Pada Jaringan Internet Dengan Metode Hierarchy Token Bucket," *TECHSI-Jurnal Teknik Informatika*, vol. 4, pp. 209-223, 2014, doi: 10.29103/techsi.v6i1.172.

[18] E. Prasetyo, A. Hamzah, E.Sutanta, "Analisa Quality of Service (Qos) Kinerja Point To Point Protocol Over Ethernet (Pppoe) Dan Point To Point Tunneling Protocol (Pptp)," *Jurnal JARKOM*, vol. 4, no. 1, pp.29-37, 2016.

[19] ETSI, "Telecommunications and Internet Protocol Harmonization Over Networks (TIPHON); General aspects of Quality of Service (QoS)," *Etsi Tr 101 329 V2.1.1*, 1999.