Optimalisasi Jaringan Komputer Menggunakan Algoritma Load Balancing di Virtual Server Citrix ADC

Hardiyan K. Ramadhan[1], Sukma Wardhana[2]
Jurusan Teknik Informatika, Fakultas Ilmu Komputer, Universitas Mercu Buana[1,2]
Jl. Raya Meruya Selatan, Kembangan, Jakarta, 11650
E-mail : 41518320034@student.mercubuana.ac.id [1], sukma@mercubuana.ac.id[2]

Abstrak -- Meningkatnya jumlah pengguna *internet* harus diimbangi dengan kesiapan kapasitas infrastruktur jaringan dan *server*. Di era digital dan merebaknya pandemi *COVID-19* memaksa semua aktifitas dilakukan secara *online*. Apabila jumlah pengguna yang mengakses *server* melebihi kapasitas infrastruktur jaringan dan *server* maka yang terjadi adalah *server down*. Diperlukan perangkat *load balancer* untuk membagi beban *traffic request* dari pengguna menuju *server*. Penelitian ini membandingkan tiga algoritma pada *load balancer Citrix ADC VPX*: *least connection, least response time*, dan *round-robin*. Menggunakan empat skenario yang berbeda pada *GNS3*, pertama adalah topologi yang hanya menggunakan *single web server*, kedua menggunakan *load balancer*, empat *web server* dan algoritma *round-robin*, ketiga algoritma diganti menjadi *least connection*, dan terakhir algoritma *least response time*. Hasil pengujian parameter *response time*, *throughput* dan *cpu utilization* menunjukan algoritma *least connection* lebih unggul. Terdapat penurunan *response time* sebesar 33%, *cpu utilization* 75% dan kenaikan *throughput* 53%. Pada parameter *service hits*, algoritma *round-robin* memiliki distribusi *traffic* yang paling merata. Maka algoritma dengan *response time*, *throughput* dan *cpu utilization* terbaik adalah *least connection.* Algoritma dengan *service hits* terbaik adalah *round-robin*. Implementasi skala besar direkomendasikan menggunakan algoritma *least connection* mengacu pada parameter *response time*, *throughput* dan *cpu utilization*. Apabila menekankan pada distribusi yang paling merata gunakan algoritma *round-robin*.

Kata Kunci: load balancing, Citrix ADC VPX, round-robin, least connection, least response time

Abstract -- The increasing number of internet users must be balanced with readiness of IT infrastructure capacity. In digital era and outbreak of COVID-19 pandemic, all activities are carried out online. If number of users accessing server exceeds IT infrastructure, server down occurs. A load balancer device is required to share traffic request load from user to server. This study compares three algorithms on Citrix ADC VPX load balancer: least connection, least response time, and round-robin. Using four different scenarios in GNS3, first is a topology uses only single web server, second uses load balancer, four web servers and round-robin algorithm, third using least connection, and the last one using least response time. The results of testing response time, throughput and CPU utilization parameters show least connection algorithm is superior. There were 33% reduction in response time, 75% CPU utilization and 53% increase in throughput. In service hits parameter, round-robin algorithm has evenest traffic distribution. So algorithm with best response time, throughput and CPU utilization is least connection. Algorithm with best service hits is round-robin. Large scale implementation is recommended using least connection algorithm with reference to response time, throughput and CPU utilization parameter. When emphasizing evenest distribution, use a round-robin algorithm.

Keywords*: *load balancing, Citrix ADC VPX, round-robin, least connection, least response time

I. PENDAHULUAN

Meningkatnya jumlah pengguna internet harus diimbangi dengan kesiapan kapasitas infrastruktur jaringan dan server. Di sisi lain, masyarakat menginginkan kecepatan akses yang maksimal [1]. Masalah muncul ketika kapabilitas *server* sudah tidak bisa menangani *request* dari *client*. Ditengah pandemi *COVID-19* saat ini semua aktifitas dilakukan dari rumah, salah satunya kegiatan perkuliahan, apabila dari segi infrastruktur *e-learning* jumlah *server* yang ada tidak dioptimisasi maka performa *server e-learning* akan menurun bahkan bisa *down*.

Sebuah perangkat *load balancer* diperlukan untuk membagi *load traffic request* dari *client* yang menuju *server*. Pada perangkat *load balancer* sendiri dibagi menjadi perangkat keras (*hardware)* dan perangkat lunak (*software/appliance). Load balancer* bekerja dengan membagi *load traffic request* dari *client* yang menuju *server*. Untuk dapat membagi *load traffic* tersebut *load balancer* bekerja menggunakan beberapa algoritma. Algoritma tersebut dibagi menjadi dua kelompok yaitu *Static Load Balancing Algorithm* dan *Dynamic Load Balancing Algorithm* [2]. Beberapa algoritma yang digunakan pada *load balancer* menurut Nguyen Xuan Phi dalam risetnya,

2017, adalah *Round-robin, Weighted Roud Robin, Least connection, Weighted Least connection,* dan *Least response time* [3]. Algoritma yang masuk kedalam *Static Load Balancing Algorithm* diantaranya *Round-robin, Central Manager, Randomized* dan *Threshold.* Sementara algoritma yang masuk kedalam *Dynamic Load Balancing Algorithm* diantaranya *Least connection, Least response time, Local Queue, dan Central Queue* [2].

Penelitian akan melakukan komparasi algoritma *least connection, least response time* dan *round-robin*. Dengan mengkaji aspek-aspek performansi jaringan yang relevan dengan jaringan *load balancing server* ketika ketiga metode ini dijalankan dengan tujuan mendapatkan algoritma yang optimal terkait pembagian beban *server* sesuai dengan kebutuhan proyek.

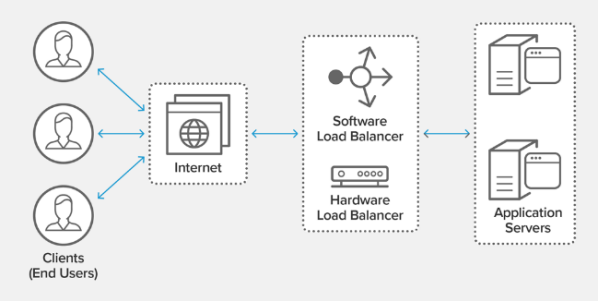
Banyak penelitian sebelumnya yang membahas mengenai implementasi *load balancing* dan komparasi algoritma didalamnya. Dalam implementasi *load balancing* sendiri perangkat yang digunakan juga beragam seperti *HA Proxy* [4], Nginx [5] [6]. Untuk komparasi algoritma sendiri pada penelitian sebelumnya yang sudah pernah dilakukan diantaranya komparasi algoritma *round-robin* dengan *least connection* [7] [4], komparasi algoritma *round-robin, least connection,* dan *ratio* [8]. Dari beberapa penelitian tersebut peneliti mencoba untuk mengangkat algoritma *least response time* sebagai salah satu algoritma yang akan diuji perfomanya dengan dikomparasi dengan algoritma *round-robin* dan *least connection* kemudian diimplementasikan pada *Citrix ADC VPX*.

Metode pengujian yang akan dilakukan dengan menggunakan *GNS3* sebagai *environment* untuk simulasi. Menggunakan empat skenario yang berbeda dengan topologi *single web server* dan *multi web server* yang terdiri dari empat *web server.*

II. TINJAUAN PUSTAKA

Load balancing merupakan sebuah teknik untuk membagi beban kerja pada 2 atau lebih *server* ketika ada *request* dari *client*, hal ini bertujuan agar *traffic* berjalan secara optimal [9].

Untuk bisa melakukan *load balancing* dibutuhkan perangkat yang disebut dengan *load balancer*. *Load balancer* bertindak sebagai *traffic cop* yang posisi dalam topologinya berada di depan *backend server farm*.



Gambar 1. Cara Kerja *Load Balancer*

Jadi fungsi *load balancer* ini untuk mendisitribusikan *request client* atau jaringan yang menuju ke banyak *server* secara efisien, memastikan *high availability* dan reliabilitas dengan mengirim *request* hanya pada *server* yang sedang *online*, dan menyediakan fleksibilitas untuk menambah atau mengurangi jumlah *server*.

Citrix Application Delivery Controller (ADC) VPX menyediakan fitur *load balancing web* dan aplikasi yang komplit, aman dan *remote* akses, akselerasi, *security*, dan fitur *offload* dalam satu set yang simpel, *virtual appliance* yang mudah di install. Organisasi IT, *cloud* dan layanan penyedia jasa telekomunikasi dapat mendeploy *Citrix ADC VPX* pada standar industri *hypervisor* kapanpun dan dimanapun dalam sebuah *datacenter.*

Citrix ADC VPX ini termasuk kedalam kategori perangkat *ADC* yang bekerja sampai *layer 7 load balancing* [10]. Karena bentuknya yang virtual berarti mengurangi biaya projek dan perawatan sehingga dapat meningkatkan profit perusahaan yang menggunakan *Citrix ADC VPX*. *Citrix ADC VPX* bisa diinstall diatas sebuah *hypervisor* atau bisa menggunakan *Citrix ADX SDX*.

Algoritma *round-robin* membagi semua *node server* akan diperlakukan setara sesuai dengan beban yang telah ditetapkan masing-masing *server*, namun tidak mengizinkan peralihan beban secara dinamis dikarenakan sifat alami beban statis. Tidak ada batasan jumlah server aktif pada *backend* [4]. *Least response time* pada dasarnya menggunakan metode *round-robin* ditambah dengan parameter untuk memilih *server* dengan koneksi paling sedikit dan memiliki *response time* paling kecil [3]. *Least connection* bekerja dengan mengirimkan *request* ke *server*, yang mana *server* yang dipilih merupakan *server* yang paling sedikit jumlah koneksinya. *Load balancer* akan memonitor jumlah koneksi *server* dan mengirimkan *request* ke *server* dengan koneksi paling sedikit [10].

Throughput, adalah *bandwidth* aktual yang terukur pada suatu ukuran waktu tertentu dalam mentransmisikan data atau *file*. *Throughput* lebih menggambarkan *bandwidth* yang sebenarnya pada suatu waktu dan pada kondisi dan jaringan tertentu yang digunakan untuk mengunduh suatu *file* dengan ukuran tertentu. [11].

Penelitian ini akan menggunakan *Response Time* dengan menggunakan standar dari *TIPHON (Telecommunications and Internet Protocol Harmonization Over Network)* dan dimasukan dalam standarisasi *delay* (Tabel 1). Nilai parameter *delay* diperoleh dengan mengelola *response time* yang terdiri dari *average, minimum,* dan *maximum* [12].

Tabel 1. Standarisasi *Delay* menurut *TIPHON*

Kategori *Delay*	***Delay***
Sangat Bagus	<150 ms
Bagus	150 s/d 300 ms
Sedang	300 s/d 450 ms
Jelek	>450 ms

Apache JMeter adalah perangkat lunak *open source*, 100% aplikasi *Java* murni dirancang untuk memuat tes perilaku fungsional dan mengukur kinerja. Ini pada awalnya dirancang untuk pengujian Aplikasi *Web* tetapi sejak diperluas untuk fungsi tes lainnya [13]. *Apache Jmeter* akan digunakan untuk mengukur perfoma paramter *cpu utilization* pada *web server*.

Apache Benchmark adalah sebuah *tool* dari *Apache organization* yang digunakan untuk mengukur performansi pada *Hypertext Transfer Protocol (HTTP) web server*. *Tool* ini digunakan untuk menghitung berapa banyak *request per second* yang dapat di layani oleh *web server* yang digunakan. Beberapa fitur dari *Apache Benchmark* seperti: *open source, simple command line, platform independent, load* dan *performance test, not extensible* [14]. *Apache Benchmark* akan digunakan untuk mengukur perfoma parameter *response time* dan *throughput web server*.

Web server merupakan perangkat lunak yang melayani permintaan *HTTP* dan *web browser* juga mengirimkan kode-kode dinamis ke *server* aplikasi. *Server* aplikasi inilah yang menerjemahkan dan memproses kode-kode dinamis menjadi kode-kode statis *HTML* dalam suatu halaman statis yang kemudian dikirimkan ke *browser* oleh *web server*. [9].

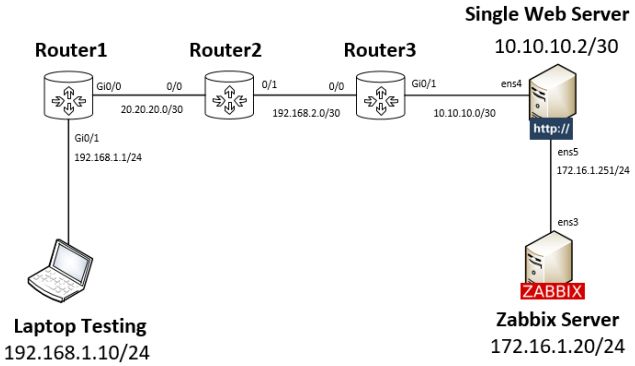
III. DESAIN PENGUJIAN

Untuk mengetahui performa *web server* sebelum dan setelah optimasi akan menggunakan dua topologi yaitu yaitu topologi *single web server* dan *multi web server* dengan *load balancer*. Topologi keduanya akan disimulasikan pada perangkat lunak *Graphical Network Simulator 3 (GNS3).* Sekilas mengenai *GNS3*, adalah perangkat lunak simulasi jaringan yang pertama kali rilis pada tahun 2008. Perangkat lunak ini dapat mengkombinasikan perangkat asli dan *virtual* untuk simulasi jaringan yang kompleks. *GNS3* banyak digunakan oleh banyak perusahaan seperti Exxon, Walmart, AT&T, dan NASA. Spesifikasi *laptop* yang menjalankan perangkat *GNS3* dan sebagai *laptop testing* atau mengacu pada Tabel 2.

Tabel 2. Spesifikasi Laptop

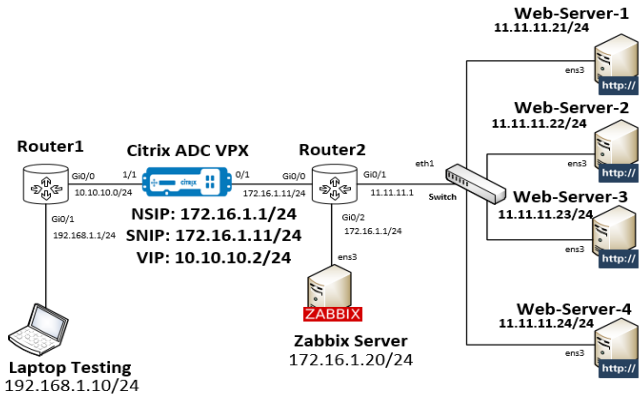
Kategori	**Spesifikasi**	**Deskripsi**
Perangkat Keras	Produsen Sistem	*Lenovo*
	Model Sistem	*Thinkpad L540*
	Processor	*Intel® Core™ i5-4300M 2.6 GHz*
	Sistem Operasi	*Xubuntu Linux 16.04*
	Memori	*16 GB*
	Media Penyimpanan	*HDD 500 GB*
Perangkat Lunak	Simulator Jaringan	*GNS3 2.2.1*
	HTTP Load Testing	*Apache Jmeter*

Berikut topologi *single web server* yang akan disimulasikan:



Gambar 2. Topologi *single web server*

Pada Gambar 2 perangkat jaringan yang digunakan adalah *router*, *web server*, *client* dan *NMS server*. Sementara itu, untuk topologi *multi web server*, berikut yang akan disimulasikan:



Gambar 3. Topologi *multi web server*

Pada Gambar 3 perangkat jaringan yang digunakan adalah *router*, *load balancer*, *web server*, *client* dan *NMS server*. Berikut spesifikasi perangkat *load balancer* mengacu pada Tabel 3.

Tabel 3. Spesifikasi *Load Balancer*

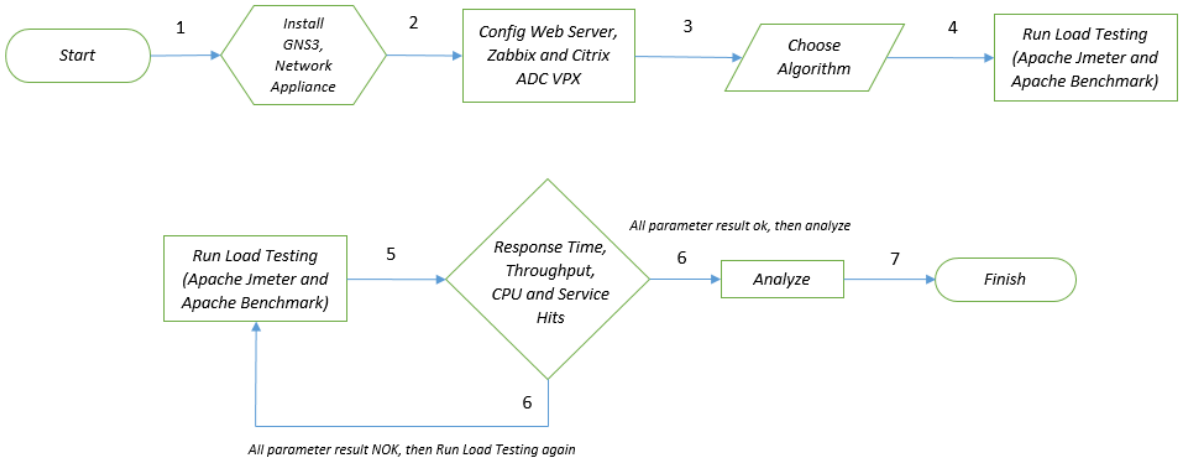
Jenis	**Spesifikasi**
vCPUs	2
RAM	4096 MB
Network Adapter	*Paravirtualized Network I/O*
Disk image	NSVPX-KVM-12.0-56.20_nc_32.qcow2
Disk interface	*Virtio*
Brand	*Citrix ADC VPX*
Version	12

Pengukuran kinerja dari ketiga algoritma (*least connection*, *least response time dan round-robin*) dilakukan dengan mengukur beberapa parameter yaitu *response time*, *throughput*, *service hits* dan *cpu utilization*. Nantinya akan dilakukan *request HTTP* dari *Laptop Testing* menggunakan software *Apache Benchmark* dan untuk *cpu utilization testing* menggunakan *Apache Jmeter*. *Service hits* didapatkan dari hasil parameter tiga algoritma yang sudah diimplementasi secara bergantian *(least connection, least response time dan round-robin)* yang muncul pada *dashboard statistics load balancing virtual server Citrix ADC VPX*.

Skenario eksperimen yang digunakan ada empat, yang pertama adalah melakukan uji beban *traffic* pada topologi *single web server*, kedua melakukan uji beban traffic pada *topologi multi web server* yang digabungkan dengan perangkat *load balancer* menggunakan Algoritma *round-robin*, ketiga merubah algoritma menjadi *least connection* dan yang terakhir merubah algoritma menjadi *least response time*. Jumlah *user* yang akan dibebankan pada keempat skenario adalah 3k, 6k, 9k, dan 12k dan menggunakan *level* 10 *concurrency*.

Tabel 4. Skenario Eksperimen

Skenario	Total *Web Server*	Algoritma	*User*
1	1	-	3k, 6k, 9k, and 12k
2	4	*Round-robin*	3k, 6k, 9k, and 12k
3	4	*Least connection*	3k, 6k, 9k, and 12k
4	4	*Least response time*	3k, 6k, 9k, and 12k



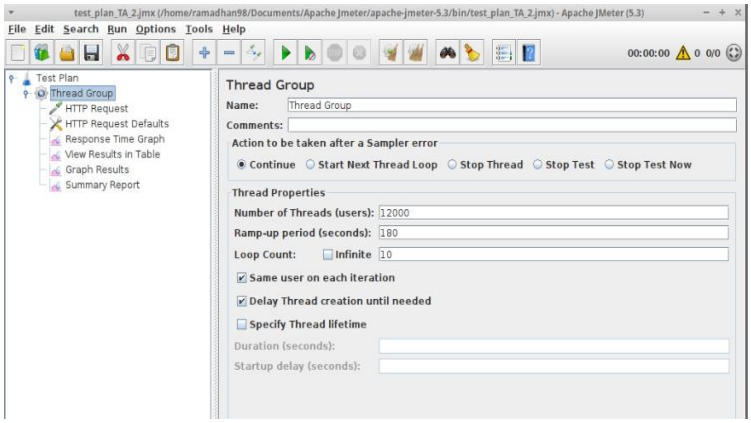
Gambar 4. Diagram *Use Case* Skenario Eksperimen (penambahan)

Skenario pertama akan menguji performa *single web server* tanpa menggunakan *load balancer*. Dengan jumlah *user* yang diuji adalah 3k, 6k, 9k, dan 12k. Parameter pertama yang diuji adalah *response time* dan *throughput* menggunakan *Apache Benchmark*. Pada Gambar 4 pengujian beban *request* 12k dengan *level concurrency* 10. Perintah yang digunakan adalah "ab -n 12000 -c 10 http://10.10.10.2/".

```
Server Software:        Apache/2.4.29
Server Hostname:        10.10.10.2
Server Port:            80

Document Path:          /
Document Length:        834 bytes

Concurrency Level:      10
Time taken for tests:   230.969 seconds
Complete requests:      12000
Failed requests:        33
   (Connect: 0, Receive: 0, Length: 33, Exceptions: 0)
Keep-Alive requests:    11866
Total transferred:      13649938 bytes
HTML transferred:       9980478 bytes
Requests per second:    51.96 [#/sec] (mean)
Time per request:       192.474 [ms] (mean)
Time per request:       19.247 [ms] (mean, across all concurrent requests)
Transfer rate:          57.71 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    0  16.2      0    1026
Processing:     0  192 455.5      8    6520
Waiting:        0  192 455.5      7    6520
Total:          0  192 455.7      8    6520
```

Gambar 5. Hasil pengujian *Apache Benchmark* 12k user pada topologi *single web server*

Selanjutnya adalah pengujian *cpu utilization* menggunakan *Apache Jmeter*. Menggunakan 12k *user* dan *ramp-up period* selama 3 menit.



Gambar 6. Pengujian *cpu utilization* menggunakan *Apache Jmeter*

Hasil *cpu utilization* dengan 12k *user* pada topologi *single web server* dilihat dari *Zabbix.* Nilai maksimum dari *cpu utilization* sebesar 8,3135 %

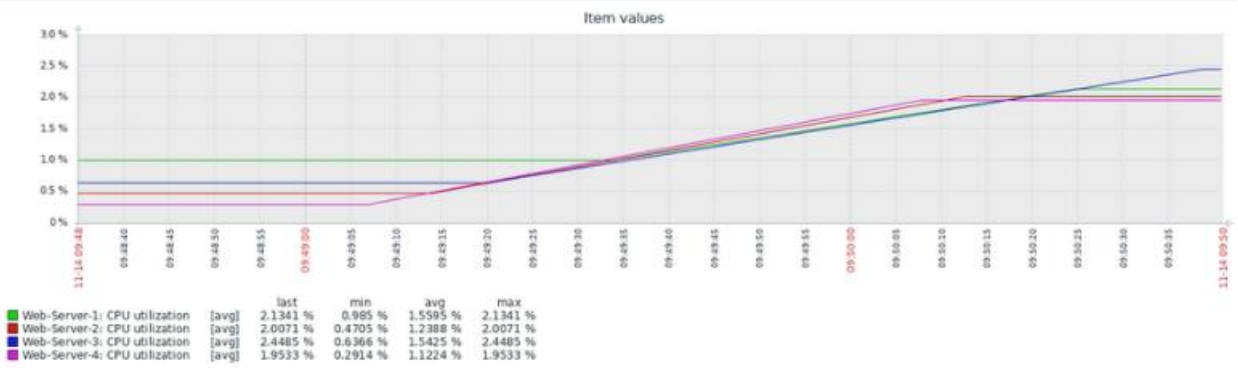
Gambar 7. Grafik *cpu utilization* topologi *single web server* dengan 12k *user*

Skenario kedua menguji performa *multi web server* dengan algoritma *round-robin* pada *load balancer Citrix ADC VPX*. Dengan jumlah *user* yang diuji adalah 3k, 6k, 9k, dan 12k. Parameter pertama yang diuji adalah *response time* dan *throughput* menggunakan *Apache Benchmark*. Pada Gambar 7 pengujian beban *request* 12K dengan level *concurrency* 10. Perintah yang digunakan adalah “ab -n 12000 -c 10 http://10.10.10.2/”.

```
Server Software:        Apache/2.4.29
Server Hostname:        10.10.10.2
Server Port:            80

Document Path:          /
Document Length:        864 bytes

Concurrency Level:      10
Time taken for tests:   159.628 seconds
Complete requests:      12000
Failed requests:        0
Total transferred:      14048598 bytes
HTML transferred:       10368000 bytes
Requests per second:    75.17 [#/sec] (mean)
Time per request:       133.024 [ms] (mean)
Time per request:       13.302 [ms] (mean, across all concurrent requests)
Transfer rate:          85.95 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    6  69.7      1    1031
Processing:     1  123 530.2      2   30140
Waiting:        1  123 530.2      2   30140
Total:          2  129 534.1      3   30141
```

Gambar 8. Hasil pengujian *Apache Benchmark* 12k *user* dengan algoritma *round-robin*

Hasil pengujian *Apache Benchmark* menunjukan 13,302 ms untuk nilai *response time* dan 85,95 Kbps untuk nilai *throughput.* Selanjutnya adalah pengujian *cpu utilization* menggunakan *Apache Jmeter*. Menggunakan 12k *user* dan *ramp-up period* selama 3 menit. Hasil *cpu utilization* dengan 12k *user* pada topologi multi *web server* dengan *algoritma round-robin* dilihat dari *Zabbix*. Nilai maksimum *dari cpu utilization* sebesar 2,4485 %.



Gambar 9. Grafik *cpu utilization* 12k *user* dengan algoritma *round-robin*

Sementara untuk hasil *service hits* pada *dashboard statistics load balancing virtual server Citrix ADC VPX* algoritma *round-robin* mengacu pada Gambar 9 dibawah ini.

Bound Service(s) Summary

Name	IP address	Port	Service type	State	Service hits
Service-1	11.11.11.21	80	HTTP	UP	3,000
Service-2	11.11.11.22	80	HTTP	UP	3,000
Service-3	11.11.11.23	80	HTTP	UP	3,000
Service-4	11.11.11.24	80	HTTP	UP	3,000

Gambar 10. *Service hits* 12k *User - Round-robin*

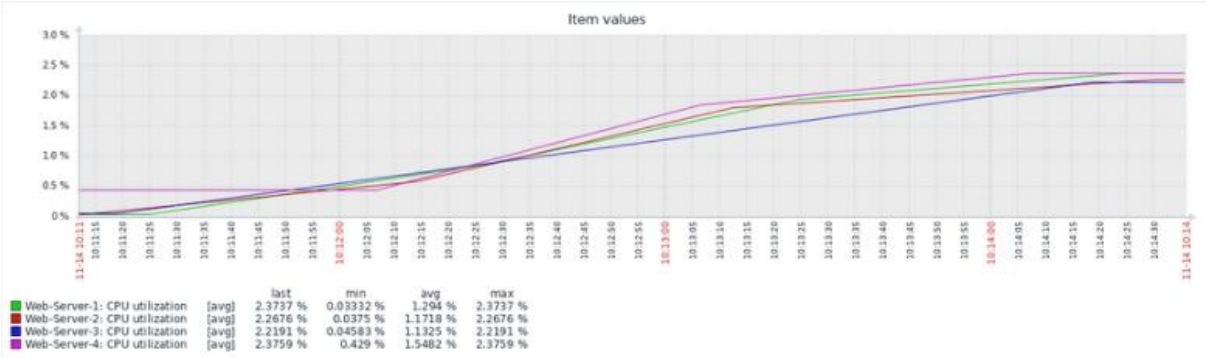
Pada gambar 9 terlihat bahwa pembagian *traffic* pada algoritma *round-robin* merata dari 12k *traffic* dibagi menjadi masing-masing 3k *traffic*.

Skenario ketiga menguji performa *multi web server* dengan algoritma *least connection* pada *load balancer Citrix ADC VPX*. Dengan jumlah *user* yang diuji adalah 3k, 6k, 9k, dan 12k. Parameter pertama yang diuji adalah *response time* dan *throughput* menggunakan *Apache Benchmark*. Pada Gambar 10 pengujian beban *request* 12K dengan *level concurrency* 10. Perintah yang digunakan adalah “ab -n 12000 -c 10 http://10.10.10.2/”.

```
Server Software:        Apache/2.4.29
Server Hostname:        10.10.10.2
Server Port:            80

Document Path:          /
Document Length:        864 bytes

Concurrency Level:      10
Time taken for tests:   155.069 seconds
Complete requests:      12000
Failed requests:        0
Total transferred:      14047487 bytes
HTML transferred:       10368000 bytes
Requests per second:    77.38 [#/sec] (mean)
Time per request:       129.224 [ms] (mean)
Time per request:       12.922 [ms] (mean, across all concurrent requests)
Transfer rate:          88.47 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    5  63.4      1    1036
Processing:     1  124 487.3      2   27546
Waiting:        1  124 487.3      2   27546
Total:          2  128 491.5      3   27547
```

Gambar 11. Hasil pengujian *Apache Benchmark* 12k user dengan algoritma *least connection*

Hasil pengujian *Apache Benchmark* menunjukan 12,922 ms untuk nilai *response time dan* 88,47 Kbps untuk nilai *throughput*. Selanjutnya adalah pengujian *cpu utilization* menggunakan *Apache Jmeter*. Menggunakan 12k *user* dan *ramp-up period* selama 3 menit. Hasil *cpu utilization* dengan 12k *user* pada topologi *multi web server* dengan algoritma *least connection* dilihat dari *Zabbix*. Nilai maksimum dari *cpu utilization* sebesar 2,3759 %.



Gambar 12. Grafik *cpu utilization* 12k *user* dengan algoritma *least connection*

Sementara untuk hasil *service hits* pada *dashboard statistics load balancing virtual server Citrix ADC VPX* algoritma *least connection* mengacu pada Gambar 12 dibawah ini.

Bound Service(s) Summary

Name	IP address	Port	Service type	State	Service hits
Service-1	11.11.11.21	80	HTTP	UP	3,165
Service-2	11.11.11.22	80	HTTP	UP	3,044
Service-3	11.11.11.23	80	HTTP	UP	2,816
Service-4	11.11.11.24	80	HTTP	UP	2,975

Gambar 13. *Service hits* 12k User - *Least connection*

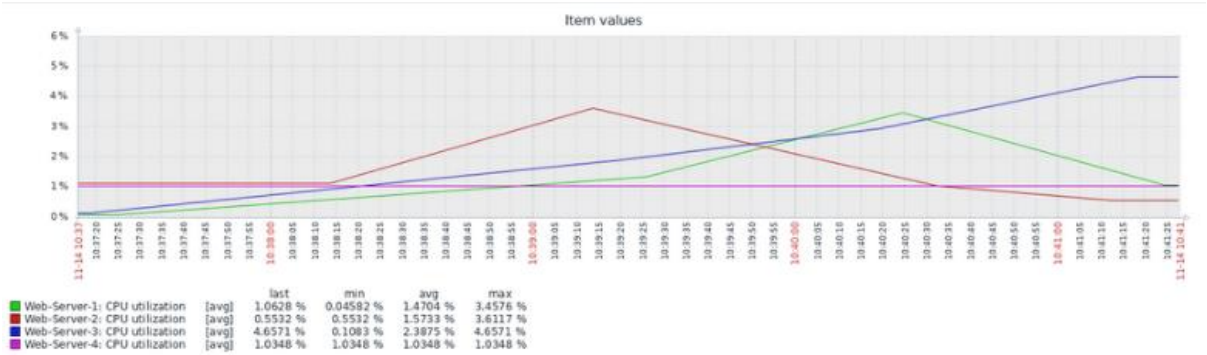
Pada Gambar 12 terlihat bahwa pembagian *traffic* pada algoritma *least connection* tidak merata, pembagian berdasarkan *web server* yang memiliki koneksi paling kecil.

Skenario keempat menguji performa multi *web server* dengan algoritma *least response time* pada *load balancer Citrix ADC VPX*. Dengan jumlah *user* yang diuji adalah 3k, 6k, 9k, dan 12k. Parameter pertama yang diuji adalah *response time* dan *throughput* menggunakan *Apache Benchmark*. Pada Gambar 13 pengujian beban *request* 12K dengan *level concurrency* 10. Perintah yang digunakan adalah “ab -n 12000 -c 10 http://10.10.10.2/”.

```
Server Software:        Apache/2.4.29
Server Hostname:        10.10.10.2
Server Port:            80

Document Path:          /
Document Length:        864 bytes

Concurrency Level:      10
Time taken for tests:   159.032 seconds
Complete requests:      12000
Failed requests:        0
Total transferred:      14047943 bytes
HTML transferred:       10368000 bytes
Requests per second:    75.46 [#/sec] (mean)
Time per request:       132.527 [ms] (mean)
Time per request:       13.253 [ms] (mean, across all concurrent requests)
Transfer rate:          86.26 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    9  90.2      1    1040
Processing:     1  121 552.9      2   32550
Waiting:        1  121 552.9      2   32550
Total:          2  130 561.0      3   32551
```

Gambar 14. Hasil pengujian *Apache Benchmark* 12k *user* dengan algoritma *least response time*

Hasil pengujian *Apache Benchmark* menunjukan 13,253 ms untuk nilai *response time dan* 86,26 Kbps untuk nilai *throughput*. Selanjutnya adalah pengujian *cpu utilization* menggunakan *Apache Jmeter*. Menggunakan 12k *user* dan *ramp-up period* selama 3 menit. Hasil *cpu utilization* dengan 12k *user* pada topologi *multi web server* dengan algoritma *least response time* dilihat dari *Zabbix*. Nilai maksimum dari *cpu utilization* sebesar 4.6571 %.



Gambar 15. Grafik *cpu utilization* 12k user dengan algoritma *least response time*

Sementara untuk hasil *service hits* pada *dashboard statistics load balancing virtual server Citrix ADC VPX* algoritma *least response time* mengacu pada Gambar 15 dibawah ini.

Bound Service(s) Summary

Name	IP address	Port	Service type	State	Service hits
Service-1	11.11.11.21	80	HTTP	UP	3,044
Service-2	11.11.11.22	80	HTTP	UP	3,915
Service-3	11.11.11.23	80	HTTP	UP	2,556
Service-4	11.11.11.24	80	HTTP	UP	2,485

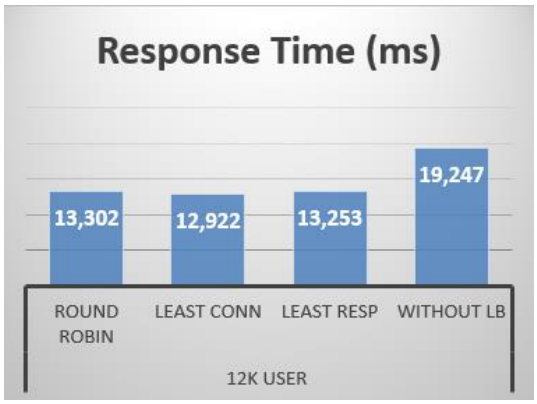
Gambar 16. *Service hits* 12k *User – Least response time*

Pada Gambar 15 terlihat bahwa pembagian *traffic* pada algoritma *least response time* tidak merata, pembagian berdasarkan *web server* yang memiliki koneksi dan *response time* paling kecil.

IV. HASIL DAN PEMBAHASAN

Setelah melakukan uji empat skenario untuk dengan menggunakan parameter *Response Time, Throughput,* dan *CPU Utilization*, berikut hasil perbandingan keempat skenario tersebut.

RESPONSE TIME



Gambar 17. Grafik Perbandingan *Response Time* 12k *User*

Pada pengujian parameter *response time*, dilakukan pada 3k, 6k, 9k, dan 12k *user*. Untuk detailnya sebagai berikut.

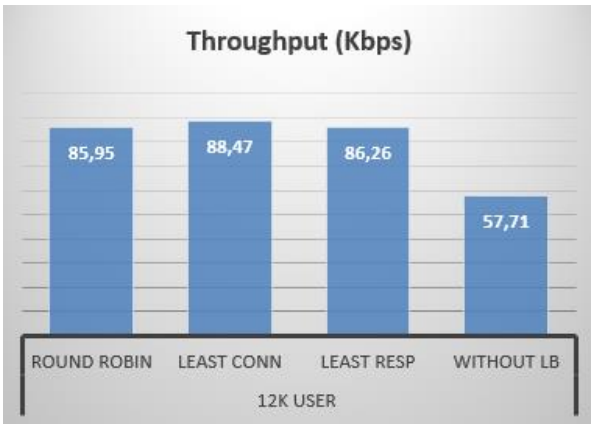
 ISSN: 2086-8251

Tabel 5. Hasil Perbandingan *Response Time*

Jumlah *User*	**Algoritma**	***Response Time* (ms)**
3k *user*	*round-robin*	13,649
	least conn	**13,548**
	least resp	14,45
	without LB	19,072
6k *user*	*round-robin*	13,701
	least conn	**13,449**
	least resp	15,047
	without LB	19,146
9k *user*	*round-robin*	13,513
	least conn	**13,041**
	least resp	14,154
	without LB	19,15
12k *user*	*round-robin*	13,302
	least conn	**12,922**
	least resp	13,253
	without LB	19,247

Dari tabel 5 bisa dilihat bahwa terdapat optimisasi *response time* setelah menggunakan *load balancer*, sementara itu algoritma *least connection* memiliki *response time* yang paling baik (terendah dalam satuan ms).

THROUGHPUT



Gambar 18. Grafik Perbandingan *Throughput* 12k *User*

Pada pengujian parameter *throughput*, dilakukan pada 3k, 6k, 9k, dan 12k user. Untuk detailnya sebagai berikut.

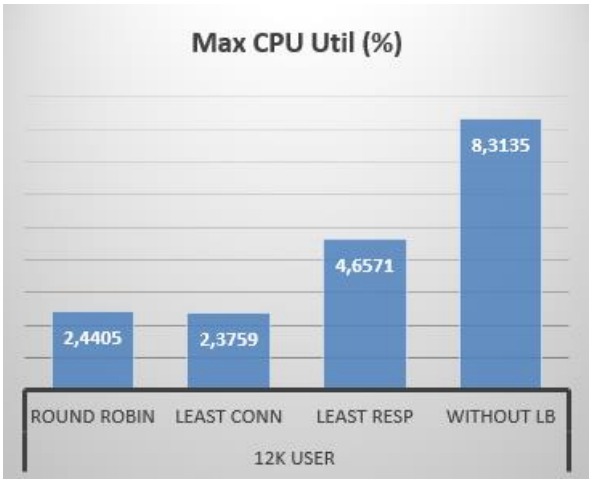
Tabel 6. Hasil Perbandingan *Throughput*

Jumlah *User*	**Algoritma**	***Throughput* (Kbps)**
3k *user*	*round-robin*	83,77
	least conn	**84,39**
	least resp	79,12
	without LB	58,39
6k *user*	*round-robin*	83,44
	least conn	**85,01**
	least resp	75,98
	without LB	58,12
9k *user*	*round-robin*	84,61

	least conn	**87,66**
	least resp	80,77
	without LB	58,06
12k *user*	*round-robin*	85,95
	least conn	**88,47**
	least resp	86,26
	without LB	57,71

Dari Tabel 6 menunjukan bahwa terdapat optimisasi *throughput* setelah menggunakan *load balancer*, sementara itu algoritma *least connection* memiliki *throughput* yang paling baik (tertinggi dalam satuan Kbps).

CPU UTILIZATION



Gambar 19. Grafik Perbandingan *Max CPU Utilization* 12k *User*

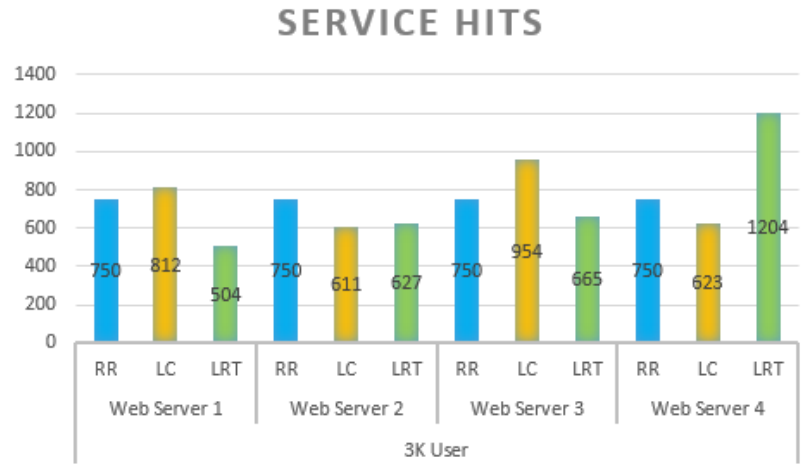
Pada pengujian parameter *cpu utilization*, dilakukan pada 3k, 6k, 9k, dan 12k *user*. Untuk detailnya mengacu pada Tabel 7.

Tabel 7. Hasil Perbandingan *CPU Utilization*

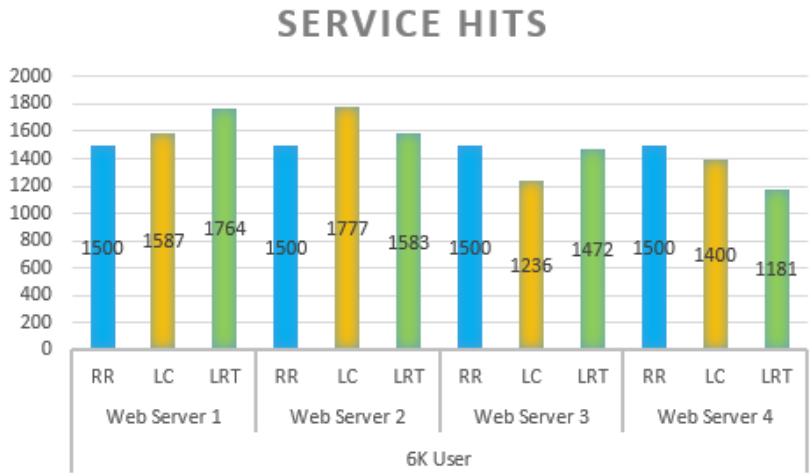
Jumlah *User*	**Algoritma**	***Max CPU Utilization* (%)**
3k *user*	*round-robin*	1,3724
	least conn	**1,2765**
	least resp	1,4042
	without LB	4,6858
6k *user*	*round-robin*	1,7452
	least conn	**1,3758**
	least resp	2,9082
	without LB	5,339
9k *user*	*round-robin*	2,3027
	least conn	**1,8411**
	least resp	2,6191
	without LB	7,2668
12k *user*	*round-robin*	2,4485
	least conn	**2,3759**
	least resp	4,6571
	without LB	8,3135

Dari Tabel 7 menunjukan bahwa terdapat optimisasi *max cpu utilzation* setelah menggunakan *load balancer*, sementara itu algoritma least connection memiliki *max cpu utilzation* yang paling baik (terendah dalam persen).

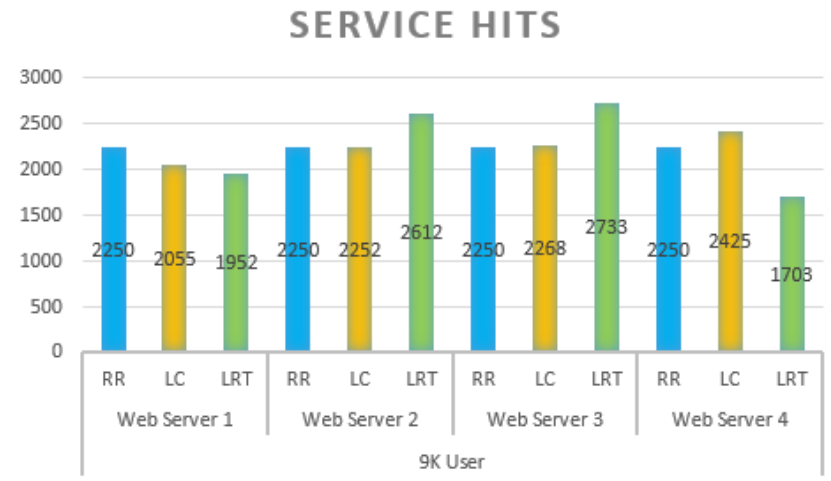
SERVICE HITS
Pada parameter *service hits* ditampilkan hasil distribusi *traffic* pada *dashboard Citrix ADC VPX* dari *laptop testing* ke *web server*. Dari sini akan telihat *behaviour* algoritma dalam membagi *traffic*.



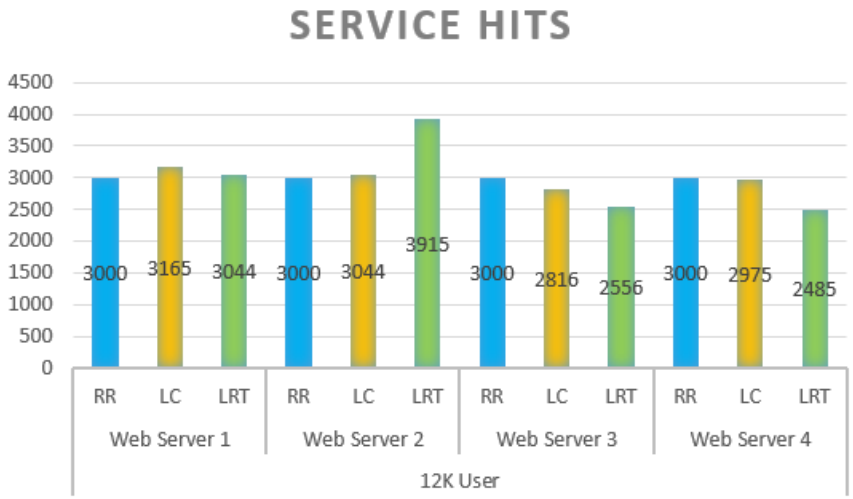
Gambar 20. Grafik Perbandingan *Service Hits* 3k *User*



Gambar 21. Grafik Perbandingan *Service Hits* 6k *User*



Gambar 22. Grafik Perbandingan *Service Hits* 9k *User*



Gambar 23. Grafik Perbandingan *Service Hits* 12k *User*

V. KESIMPULAN

Terdapat optimisasi paramater *response time, throughput,* dan *cpu utilization* setelah menggunakan *load balancer* dan perubahan jumlah *web server*. Adapun algoritma yang performanya paling baik menurut dua parameter yang diuji adalah *least connection* dengan penurunan *response time* sebesar 33%, *cpu utilization* sebesar 75% dan kenaikan *throughput* sebesar 53%. Sementara pada parameter *service hits***,** algoritma *round-robin* memiliki distribusi *traffic* yang merata dibanding algoritma lain. Algoritma *least response time* dalam pengujian empat parameter secara *overall* menunjukan performa dibawah algoritma *round-robin* dan *least-connection.*

Pengujian yang menggunakan empat jumlah user masing-masing 3k, 6k, 9k, dan 12k menghasilkan hasil yang konsisten dengan algoritma *round-robin* selalu rata dalam mendistribusikan *traffic,* sementara *least connection* paling baik pada performa *response time, throughput,* dan *cpu utilization.*

Pada implementasi skala besar apabila terdapat kebutuhan untuk membagi rata *traffic* ke semua *back-end server* yang ada disarankan untuk menggunakan algoritma *round-robin*, sementara itu untuk kebutuhan yang berfokus kepada performa parameter *response time, throughput,* dan *cpu utilization* disarankan menggunakan *least connection.*

VI. DAFTAR PUSTAKA

[1] F. P. Perdana, B. Irawan, R. Latuconsina, F. Teknik, U. Telkom, and L. Balancing, “Analisis Performansi Load Balancing Dengan Algoritma Weighted Round Robin Pada Software Defined Network (SDN),” *e-Proceeding Eng.*, vol. 4, no. 3, pp. 4161–4168, 2017.

[2] P. Beniwal, “A comparative study of static and dynamic Load Balancing Algorithms,” *Int. J. Adv. Res.*

Comput. Sci. Manag. Stud., vol. 2, no. 12, pp. 386–392, 2014.

[3] N. Xuan Phi and T. Cong Hung, "Load Balancing Algorithm to Improve Response Time on Cloud Computing," *Int. J. Cloud Comput. Serv. Archit.*, vol. 7, no. 6, pp. 01–12, 2017, doi: 10.5121/ijccsa.2017.7601.

[4] 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.

[5] W. Chen, A. Noertjahyana, and J. Andjarwirawan, "Analisis Perbandingan Kinerja Algoritma Load Balancer NGINX pada Studi Kasus PRS," *J. Infra*, vol. 7, no. 2, pp. 60–64, 2019.

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

[7] A. Nugroho, W. Yahya, and Kasyful Amron, "Analisis Perbandingan Performa Algoritma Round Robin dan Least Connection untuk Load Balancing pada Software Defined Network," *J. Pengemb. Teknol. Inf. dan Ilmu Komput.*, vol. 1, no. 12, pp. 1568–1577, 2017, [Online]. Available: http://j-ptiik.ub.ac.id.

[8] H. Nasser and T. Witono, "Analisis Algoritma Round Robin, Least Connection, Dan Ratio Pada Load Balancing Menggunakan Opnet Modeler," *J. Inform.*, vol. 12, no. 1, pp. 25–32, 2016, doi: 10.21460/inf.2016.121.455.

[9] D. Rahmana, R. Primananda, and W. Yahya, "Analisis Load Balancing Pada Web Server Menggunakan Algoritme Weighted Least Connection," *J. Pengemb. Teknol. Inf. dan Ilmu Komput. Univ. Brawijaya*, vol. 2, no. 3, pp. 915–920, 2017, doi: 10.1016/0028-3932(72)90008-5.

[10] I. Ivanisenko, "Methods and Algorithms of Load Balancing," *Int. J. "Information Technol. Knowledge,"* vol. 9, no. 4, pp. 340–375, 2015.

[11] Hasanul Fahmi, "Analisis Qos (Quality of Service) Pengukuran Delay, Jitter, Packet Lost Dan Throughput Untuk Mendapatkan Kualitas Kerja Radio Streaming Yang Baik," *J. Teknol. Inf. dan Komun.*, vol. 7, no. 2, pp. 98–105, 2018.

[12] M. F. W. Simanjuntak, O. D. Nurhayati, and E. D. Widianto, "Analisis Quality of Service (QoS) Jaringan Telekomunikasi High-Speed Downlink Packet Access (HSDPA) pada Teknologi 3.5G," *J. Teknol. dan Sist. Komput.*, vol. 4, no. 1, p. 67, 2016, doi: 10.14710/jtsiskom.4.1.2016.67-76.

[13] D. I. Permatasari, "Pengujian Aplikasi menggunakan metode Load Testing dengan Apache JMeter pada Sistem Informasi Pertanian," *J. Sist. dan Teknol. Inf.*, vol. 8, no. 1, p. 135, 2020, doi: 10.26418/justin.v8i1.34452.

[14] A. Y. Chandra, "Analisis Performansi Antara Apache & Nginx Web Server Dalam Menangani Client Request," *J. Sist. dan Inform.*, vol. 14, no. 1, pp. 48–56, 2019, doi: 10.30864/jsi.v14i1.248.