Journal of Informatics and Computer Engineering Research (JICER) ISSN: 3089-6320
Vol. 1, No. 2, Desember 2024, hlm. 42-49

EVALUASI KINERJA LOAD BALANCING DENGAN ALGORITMA SCHEDULLING NEVER QUEUE

Zulfianndari, Irmawati, Rini Nur
Teknik Informatika dan Komputer, Politeknik Negeri Ujung Pandang, Indonesia

Info Artikel

Riwayat artikel:
Received, (1 Desember 2024)
Revised, (15 Desember 2024)
Accepted, (22 Desember 2024)

Kata kunci:
Load balancing;
Schedulling algoritma;
Never Queue;
Round Robin.

ABSTRACT

Load balancing is used as a technique to handle large loads that cannot be carried out by a single server, so that the server does not experience overload. In handling load sharing, Load balancing uses a scheduling algorithm (Scheduling). The scheduling algorithm that is generally used is Round Robin which works by dividing requests evenly and then creating a queue for the server so that unfinished processes wait in the queue for quite a long time. In the Load balancing system there is an algorithm that adopts two speed models, which works by looking at the server status and the smallest connection delay, namely the Never Queue Algorithm. This study aims to determine the performance of Load balancing when the Schedulling Never Queue Algorithm is applied, based on predetermined scenarios and parameters. This study succeeded in implementing the Never Queue Algorithm in a Load balancing system for the Apache web server where the Time Per Request value will be lower if the Request received is larger when compared to using the Round Robin Algorithm. The Request Per Second value increases when the Requests sent are getting bigger. In terms of sharing server connections, the load balancer will share the load on the number of Requests based on the Shortest Expected Delay (SED) Algorithm, so several web servers receive different numbers of connections, so that processes don't stay in queues for a long time.

ABSTRAK

Load balancing digunakan sebagai teknik untuk menangani beban yang besar yang tidak dapat dilakukan oleh server tunggal, tujuannya agar server tidak mengalami overload. Dalam menangani pembagian beban, *Load balancing* menggunakan algoritma penjadwalan *(Schedulling)*. Algoritma penjadwalan yang umumnya digunakan adalah Round Robin yang berkerja dengan cara membagi rata *request* kemudian membuat antrian untuk servernya sehingga proses yang belum selesai menunggu dalam antrian dalam waktu yang cukup lama. Pada sistem *Load balancing* terdapat algoritma yang mengadopsi dua model kecepatan, yang bekerja dengan melihat status server dan *delay* koneksi terkecil, yaitu Algoritma Never Queue. Penelitian ini bertujuan untuk mengetahui kinerja *Load balancing* ketika diterapkan Algoritma Schedulling Never Queue, berdasarkan skenario dan parameter yang telah ditentukan. Penelitian ini berhasil menerapkan Algoritma Never Queue dalam sistem *Load balancing* untuk *Apache web server* dimana nilai *time per request* akan lebih rendah jika *request* yang diterima lebih besar jika dibandingkan dengan menggunakan Algoritma Round Robin. Nilai *request per second* semakin meningkat di saat *request* yang dikirimkan semakin besar. Dalam hal pembagian koneksi server, *load balancer* akan melakukan pembagian beban jumlah *request* berdasarkan Algoritma Shortest Expected Delay (SED), maka beberapa *web server* menerima koneksi dalam jumlah yang berbeda, sehingga proses tidak berada dalam antrian dalam waktu yang lama.

Penulis Korespondensi:
Irmawati
Teknik Informatika dan Komputer, Politeknik Negeri Ujung Pandang, Jl. Perintis Kemerdekaan KM. 10, Makassar
Email: irmawati@poliupg.ac.id

1. PENDAHULUAN

Kemajuan teknologi saat ini terus mengalami perkembangan seiring dengan semakin meningkatnya pengguna internet. Hal ini didukung dengan hasil survei yang telah dilakukan oleh Asosiasi Penyelenggara Jaringan Internet Indonesia (APJII), pada periode 2021-2022 terdapat 210 juta pengguna internet dari total 272 juta jiwa penduduk Indonesia. Tingkat penetrasi internet di Indonesia mencapai 77.02%, sehingga jumlah tersebut meningkat sebesar 6,78% dibandingkan dengan periode sebelumnya. Sekitar 80 persen dari pengguna internet mengakses informasi melalui situs web yang berhubungan dengan perusahaan, instansi ataupun institusi pendidikan dan perekonomian [1]. Situs web tersebut tentu menerima ratusan koneksi dari *client* dalam satu waktu.

Request dalam jumlah yang besar pada sebuah server dapat mengakibatkan server *down* karena *overload* dengan banyaknya *request* yang diterima. Biasanya dilakukan *upgrade* atau menambah server untuk mengatasi hal ini. Namun, cara tersebut membutuhkan biaya yang besar. Maka dari itu dibutuhkan suatu sistem server yang dapat mengatasi sejumlah akses yang tinggi [2].

Load balancing adalah teknik untuk pendistribusian trafik jaringan atau beban kerja ke beberapa sumber daya untuk menghindari kelebihan beban pada salah satu sumber daya [3]. *Load balancing* tidak hanya dapat dilakukan pada perangkat jaringan seperti router, switch dan lain-lain, melainkan dapat juga dilakukan pada server. *Load balancing* pada server digunakan untuk membagi beban kerja ke beberapa server. Hal ini dilakukan untuk mengatasi server yang sedang mengalami kelebihan beban yang membuat server tidak mampu menampung dan menerima *request* yang berlebih dari pengguna *(overload).*

Teknik *Load balancing* dapat dijadikan solusi untuk mengatasi server down ketika mendistribusikan beban kerja dengan mempertimbangkan kapasitas setiap server [4]. Selain itu, Teknik in juga untuk memastikan bahwa salah satu server tidak menanggung terlalu banyak beban *request* [5]. Proses pembagian beban pada sistem *load balancing* memiliki teknik dan algoritma tersendiri. Perangkat *load balancing* yang kompleks biasanya menyediakan berbagai teknik dan algoritma penjadwalan pembagian beban, tujuannya untuk menyesuaikan pembagian beban dengan karakteristik dari masing-masing pengguna internet [6].

Algoritma penjadwalan pembagian beban tradisional pada *load balancing* yang paling umum digunakan adalah Round Robin [7]. Algoritma Round Robin padal*Load balancing* merupakan algoritma paling sederhana dan banyak digunakan oleh perangkat *load balancing.* Algoritma ini membagi beban dengan cara bergiliran dan berurutan dari satu server ke server lainnya, sehingga membentuk perputaran [8]. Setiap proses pada Algoritma Round Robin akan mengalami eksekusi dan *pause* yang sama berdasarkan *quantum time* yang diberikan serta tidak ada prioritas tertentu dalam algoritma ini. Sebaliknya, apabila *quantum time* terlalu besar, maka *response* time dari proses-proses yang terkait akan semakin tinggi [9]. sehingga proses yang belum selesai akan menunggu dalam antrian dalam waktu yang cukup lama, karena Algoritma Round Robin bekerja dengan mengantrikan servernya.

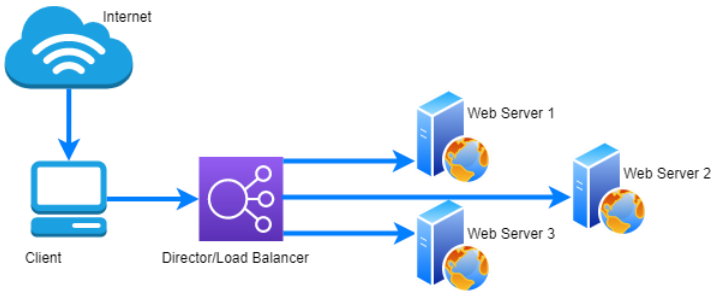
Untuk mengatasi hal tersebut, diperlukan algoritma penjadwalan yang diharapkan mampu meningkatkan kinerja *load balancing*. Salah satu Algoritma *load balancing* yang dikenal dengan kecepatannya dalam pembagian beban adalah Algoritma Never Queue. Ketika ada server *idle* yang tersedia, maka *request* akan dikirim ke server *idle* tersebut. Sebaliknya, jika tidak ada server dalam kondisi *idle* maka *request* akan ditangani oleh Algoritma Shortest Expected Delay (SED) dengan mengirimkan *request* kepada server yang memiliki estimasi terhadap *delay* dengan koneksi terpendek [10], sehingga *request* yang diterima tidak harus menunggu di dalam antrian dengan waktu yang lama.

Berdasarkan latar belakang tersebut, maka pada penelitian ini dilakukan penerapan Algoritma Schedulling Never Queue untuk menganalisis kinerja *load balancing* yang diterapkan pada Linux Virtual Server yang digunakan pada web server. Dengan menggunakan beberapa skenario pengiriman jumlah *request* yang diharapkan dapat memberikan hasil kinerja *load balancing* yang baik berdasarkan parameter yang ditentukan, sehingga dapat dilihat kinerja *load balancing* ketika diterapkan Algoritma Never Queue.

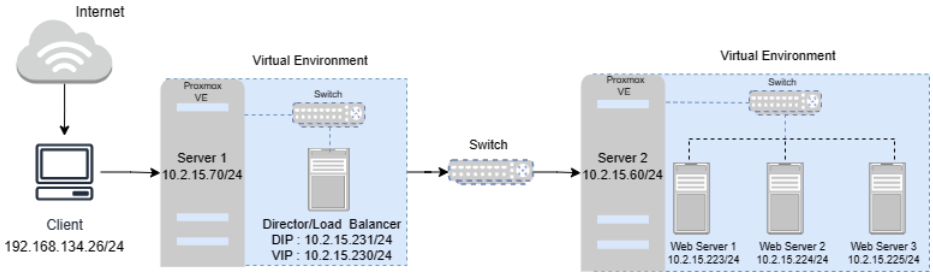
2. METODE

Metode penelitian yang digunakan dalam penelitian ini menggunakan tahapan sebagai berikut seperti yang ditunjukkan pada gambar 1.

1. Studi pendahuluan dilakukan dengan menggunakan penelitian terapan yang bertujuan untuk memberikan solusi terhadap suatu permasalahan tertentu dan hasil yang didapatkan, bisa langsung diterapkan untuk mengatasi permasalahan yang terjadi.
2. Analisis kebutuhan bertujuan untuk memperoleh semua kebutuhan yang diperlukan untuk tujuan penelitian.
3. Desain sistem dapat dilihat pada gambar 2 dan gambar 3.



Gambar 1. Topologi Sistem



Gambar 2. Detail Topologi Penelitian

Berdasarkan gambar topologi di atas, *request* dari client akan didistribusikan ke dalam tiga server web, yaitu web server 1, web server 2 dan web server 3. *Request* yang dikirimkan client akan diterima oleh web server berdasarkan pemilihan yang dilakukan oleh *server load balancing*. Proses pemilihan web server tersebut ditentukan oleh server *Load balancing* berdasarkan proses kerja algoritma penjadwalan *Load balancing* yang telah diterapkan pada sistem *Load balancing* tersebut.

4. Instalasi dan Konfigurasi
   Tahap ini merupakan tahapan awal penerapan sistem dengan tujuan agar sistem dapat beroperasi dengan baik dan memudahkan dalam menjalankan skenario pengujian pengambilan data analisis. Pada tahapan ini dilakukan instalasi *software* dan konfigurasi sistem.
5. Pengujian dan Analisis Data
   Setiap skenario dilakukan dengan beberapa jenis pengujian yaitu pengujian *Time Per Request, Request Per Second* dan jumlah koneksi yang diterima masing-masing server berdasarkan algoritma yang diterapkan pada *load balancing*. Pengujian jumlah *client* yang diuji berjumlah 1000 dan jumlah *request* yang dikirimkan dimulai dari 1500, 5500, 10500 dan 15000. Pengujian dilakukan sebanyak 3 kali dengan mengambil nilai rata-rata dari tiap percobaan dengan jeda waktu percobaan 300 *second* tiap percobaan.

3. HASIL DAN PEMBAHASAN

3.1. Pengujian

Pada tahapan ini akan dilakukan pengujian terhadap sisteml *load balancing* menggunakan *tools* Apache Benchmark berdasarkan skenario dan parameter yang telah ditentukan. Hasil dari pengujian ini akan dianalisis hingga mendapatkan kesimpulan hasil kinerja *load balancing* ketika diterapkan Algoritma Never Queue, dimana sebelumnya sistem *load balancing* menggunakan Algoritma Round Robin.

3.2. Pengujian sistem *load balancing* dengan menerapkan algortima Round Robin

Dengan menggunakan menggunakan *tools* Apache Bench seperti pada gambar 3, dapat terlihat bahwa jika client mengirimkan sebesar 200 dengan jumlah *concurrency client* 10 maka nilai time per *request* sebesar 128.657 ms dan nilai *Request* Per Second sebesar 7.77 second.

```
Server Software:        Apache/2.4.52
Server Hostname:        10.2.15.230
Server Port:            80

Document Path:          /
Document Length:        Variable

Concurrency Level:      10
Time taken for tests:   25.731 seconds
Complete requests:      200
Failed requests:        0
Total transferred:      2182731 bytes
HTML transferred:       2127931 bytes
Requests per second:    7.77 [#/sec] (mean)
Time per request:       1286.569 [ms] (mean)
Time per request:       128.657 [ms] (mean, across all concurrent requests)
Transfer rate:          82.84 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:       44  457 207.2    450    1575
Processing:    62  696 955.5    522   11308
Waiting:       61  637 621.2    522    6132
Total:        106 1153 1010.1   1029   12087

Percentage of the requests served within a certain time (ms)
  50%   1029
  66%   1069
  75%   1158
  80%   1203
  90%   1403
  95%   1868
  98%   3591
  99%   5628
 100%  12087 (longest request)
```

Gambar 3. Hasil Pengujian *load balancing* Round Robin.

Kemudian untuk pembagian koneksi kepada server dapat terlihat dengan menggunakan ipvsadm pada gambar 4 di bawah ini:

```
IP Virtual Server version 1.2.1 (size=4096)
Prot LocalAddress:Port Scheduler Flags
  -> RemoteAddress:Port           Forward Weight ActiveConn InActConn
TCP  10.2.15.230:80 rr
  -> 10.2.15.227:80               Route   1      0          67
  -> 10.2.15.228:80               Route   1      0          67
  -> 10.2.15.229:80               Route   1      0          66
```

Gambar 4. Pembagian Jumlah Koneksi *load balancing* Round Robin

3.2.1 Pengujian sistem *load balancing* dengan menerapkan algortima Never Queue

Pada gambar 5 dapat dilihat client mengirimkan *Request* sebesar 200 *request* dengan jumlah *concurrency client* 10 menggunakan *tools* Apache Bench, hasil yang dapatkan nilai *time per request* sebesar 99.047 ms dan nilai *reques*t per second sebesar 10.10 *second.*

```
Server Software:        Apache/2.4.52
Server Hostname:        10.2.15.230
Server Port:            80

Document Path:          /
Document Length:        Variable

Concurrency Level:      10
Time taken for tests:   19.809 seconds
Complete requests:      200
Failed requests:        0
Total transferred:      2182737 bytes
HTML transferred:       2127937 bytes
Requests per second:    10.10 [#/sec] (mean)
Time per request:       990.474 [ms] (mean)
Time per request:       99.047 [ms] (mean, across all concurrent requests)
Transfer rate:          107.60 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:       51  435 194.7    449     816
Processing:    52  528 205.5    539     918
Waiting:       52  524 205.5    538     918
Total:        104  962 279.1   1041    1357

Percentage of the requests served within a certain time (ms)
  50%   1041
  66%   1061
  75%   1080
  80%   1100
  90%   1151
  95%   1179
  98%   1328
  99%   1340
 100%   1357 (longest request)
```

Gambar 5. Hasil Pengujian *load balancing* Never Queue

Kemudian untuk pembagian koneksi kepada server dapat dilihat pada gambar 6 di bawah ini:

```
IP Virtual Server version 1.2.1 (size=4096)
Prot LocalAddress:Port Scheduler Flags
  -> RemoteAddress:Port           Forward Weight ActiveConn InActConn
TCP  10.2.15.230:80 nq
  -> 10.2.15.227:80               Route   1      0          79
  -> 10.2.15.228:80               Route   1      0          63
  -> 10.2.15.229:80               Route   1      0          58
```

Gambar 6. Pembagian Jumlah Koneksi *load balancing* Never Queue

3.3. Hasil Pengujian

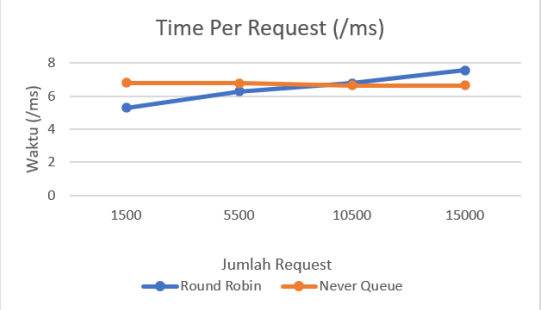
Hasil dari pengujian yang telah diperoleh dari setiap skenario. Pada rangkuman hasil penelitan, dapat dilihat bagaimana kinerja *load balancing* ketika diterapkan algoritma yang berbeda berdasarkan skenario dan parameter yang telah ditentukan.

3.3.1 Pengujian *Time Per Request*

Pengujian *Time Per Request* dilakukan dengan tujuan mengetahui waktu yang dibutuhkan untuk mendapatkan respon terhadap satu *request* yang dikirimkan. Berikut hasil pengujian *time per request* yang telah dilakukan pada *load balancing* dengan menerapkan dua algoritma yang berbeda, yaitu Algoritma Round Robin dan Never Queue.

Tabel 1. Hasil *Time Per Request* Round Robin dan Never Queue

Concurrency (Jumlah Client)	Jumlah Request	Request Per Second (/ms)	
		Round Robin	Never Queue
10000	1500	5280	6788
	5500	6264	6779
	10500	6779	6647
	15000	7545	6636



Gambar 7. Hasil *Time Per Request* Round Robin dan Never Queue

Hasil *time per Request* yang dihasilkan *load balancing* dengan menerapkan Algoritma Never Queue, nilai *time per Request* yang didapatkan cenderung menurun dan lebih rendah dibandingkan dengan *load balancing* yang menerapkan Algoritma Round Robin. Dari hasil *request* yang dilihat pada tabel 1 dan gambar 7 *load balancing* menggunakan Algoritma Never Queue menghasilkan waktu respon yang akan lebih rendah jika *request* yang diterima lebih banyak. Algoritma Never Queue menunjukkan stabilitas nilai *time per request* yang signifikan. Hal ini terjadi karena pada Algoritma Never Queue tidak mengandalkan peratan *request*, Never Queue akan mengarahkan *request* berdasarkan kondisi server, dimana ada server yang berstatus *idle* atau mempunyai *delay* koneksi terpendek, maka *request* akan diarahkan ke server tersebut, sehingga hal ini mencegah terjadinya antrian yang besar jika *request* yang diterima semakin banyak.

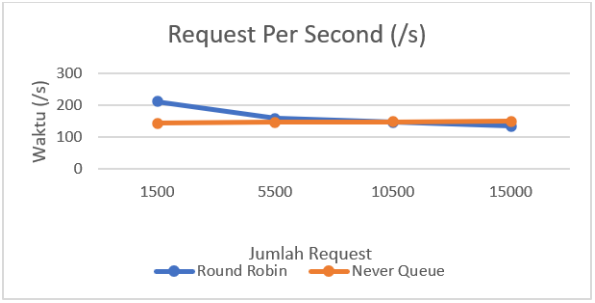
Hasil *time per request* yang dihasilkan *load balancing* dengan menerapkan Algoritma Round Robin cenderung lebih tinggi dibanding *load balancing* yang menerapkan Algoritma Never Queue. Dari hasil *request* yang dilihat pada table 1 dan gambar 7 *load balancing* menggunakan Algoritma Round Robin menghasilkan waktu respon yang akan lebih tinggi jika *request* yang diterima lebih banyak. Hal ini terjadi karena pada Algoritma Round Robin mengandalkan perataan *request* yang didapat pada setiap server, dalam hal ini tidak ada proses yang diprioritaskan. Maka antrian yang nantinya akan terjadi juga akan semakin besar jika *request* yang diterima semakin besar.

3.3.2 Pengujian *Request Per Second*

Pengujian *Request Per Second* dilakukan dengan tujuan mengetahui seberapa banyak *request* yang dilayani dalam satu waktu. Berikut hasil pengujian *Request Per Second* yang telah dilakukan pada *load balancing* dengan menerapkan dua algoritma yang berbeda, yaitu Algoritma Round Robin dan Never Queue

Tabel 2. Hasil *Request* Per Second Round Robin dan Never Queue

Concurrency (Jumlah Client)	Jumlah Request	Request Per Second (/s)	
		Round Robin	Never Queue
10000	1500	179,36	144,37
	5500	159,71	147,66
	10500	147,78	148,56
	15000	134,53	150,26



Gambar 8. Hasil *Request* Per Second Round Robin dan Never Queue

Berdasarkan *load balancing* yang menerapkan Algoritma Never Queue, nilai Request Per Second yang didapatkan cenderung meningkat dan lebih tinggi dibandingkan dengan *load balancing* yang menerapkan Algoritma Round Robin. Dari hasil *request* yang dilihat pada tabel 2 dan gambar 8 *load balancing* menggunakan Algoritma Never Queue menghasilkan menghasilkan pelayanan *request* yang semakin tinggi, meskipun *request* yang diterima semakin banyak. Hal ini terjadi karena Algoritma Never Queue tidak melakukan perataan *request*, melainkan Algoritma ini bekerja dengan mengadopsi dua model kecepatan, dimana ada server yang berstatus *idle* maka *request* akan diarahkan ke server tersebut, sebaliknya jika tidak ada server dalam kondisi *idle* maka perhitungan *Request* akan dilakukan menggunakan Algoritma shortest expected delay (SED) dengan mengirimkan *Request* kepada server yang memiliki estimasi terhadap *delay* dengan koneksi terpendek. Sehingga *Request* yang diterima tidak harus menunggu dalam antrian dengan waktu yang lama.

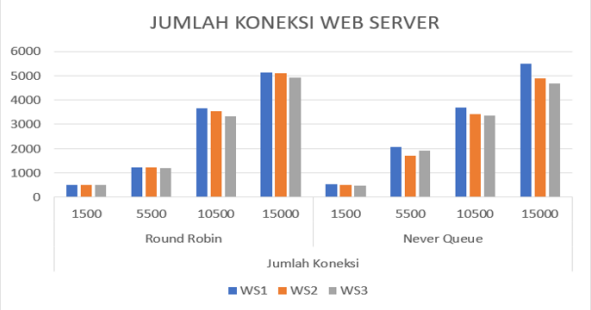
Hasil *Request Per Second kload balancing* dengan menerapkan Algoritma Round Robin cenderung lebih rendah dibanding *load balancing* yang menerapkan Algoritma Never Queue. Dari hasil *Request* yang dilihat pada tabel 2 dan gambar 8 *load balancing* menggunakan Algoritma Round Robin menghasilkan pelayanan *request* yang semakin rendah jika *request* yang diterima semakin banyak. Hal ini terjadi karena Algoritma Round Robin bekerja dengan mengantrikan servernya, dalam hal ini Algoritma Round Robin akan menempatkan proses-proses yang belum selesai menunggu dalam antrian dalam waktu yang cukup lama, sebab Round Robin bekerja dengan mengantrikan servernya, sehingga *request* harus menunggu.

3.3.3 Jumlah Koneksi

Pengujian ini dilakukan untuk melihat bagaimana pembagian koneksi yang dilakukan *Director* terhadap *web* server. Pada pengujian ini dapat dilihat berapa banyak jumlah *request* yang diterima masing-masing server berdasarkan pemilihan algoritma penjadwalan yang telah diterapkan pada *load balancing*. Berikut hasil pengujian pembagian jumlah koneksi yang telah dilakukan pada *load balancing* dengan menerapkan dua algoritma yang berbeda, yaitu Algoritma Round Robin dan Never Queue.

Tabel 3. Hasil Pembagian Jumlah Koneksi Round Robin dan Never Queue

Concurrency (Jumlah Client)	Jumlah Request	Request Per Second (/s)					
		Round Robin			Never Queue		
		WS1	WS2	WS3	WS1	WS2	WS3
10000	1500	506	501	492	522	492	486
	5500	1229	1220	1194	2063	1694	1908
	10500	3647	3530	3328	3701	3408	3366
	15000	5127	5106	4924	5486	4884	4669



Gambar 9. Hasil Pembagian Jumlah Koneksi Round Robin dan Never

Pada tabel 3 dan gambar 9 dapat dilihat pembagian jumlah koneksi pada masing-masing *web* server yang dilakukan oleh *load balancing* yang diterapkan Algoritma Never Queue, menghasilkan pembagian jumlah koneksi yang berbeda terhadap ke tiga *web* server. Jika dilihat pada tabel 3 dan gambar 9 *web* server 1 selalu mendapat jumlah koneksi yang lebih besar dibanding *web* server 2 dan *web* server 3. Hal ini disebabkan Algoritma Never Queue mempunyai server prioritas dalam hal ini menjadikan *web* server 1 sebagai server prioritas pertama, sehingga ketika *web* server 1 belum berstatus *idle*, artinya prioritas pengiriman koneksi akan di arahkan ke server 2. Namun jika pada keadaan dimana ke tiga server masih memproses koneksi, maka penentuan koneksi dihitung dari harapan *delay* terpendek dengan menggunakan Algoritma shortest expected Delay (SED) sehingga koneksi selanjutnya akan di arahkan ke server lain yang memiliki harapan delay terpendek. Oleh karena itu banyaknya *Request* yang diterima masing-masing *web* server pada *load balancing* yang telah diterapkan Algoritma Never Queue menerima *request* dalam jumlah yang berbeda. Hal ini disebabkan Algoritma Never Queue membagi koneksi dengan mengadopsi dua model kecepatan. Ketika ada server *idle* yang tersedia, maka *request* akan dikirim ke server *idle* tersebut. Sebaliknya, jika tidak ada server dalam kondisi *idle* maka *Request* akan ditangani oleh Algoritma Shortest Expected Delay (SED).

Kemudian dapat dilihat pembagian jumlah koneksi pada masing-masing *web* server yang dilakukan oleh *load balancing* yang diterapkan Algoritma Round Robin, menghasilkan pembagian jumlah koneksi yang seimbang dibanding *Load balancing* yang diterapkan Algoritma Never Queue. Ketiga *web* server menerima *request* dalam jumlah yang sama atau relatif sama. Hal ini terjadi karena Algoritma Round Robin menganggap semua server sama atau menyamakan semua servernya, dalam hal ini Round Robin tidak memiliki prioritas dalam menjalankan proses *Request*. Sehingga semua server mendapatkan beban yang sama.

4. KESIMPULAN

1. Penelitian ini berhasil menerapkan Algoritma Never Queue pada *load balancing* sehingga dapat diketahui kinerja *load balancing* yang dihasilkan berdasarkan skenario dan parameter yang telah ditentukan.
2. Data hasil pengujian yang diperoleh berdasarkan skenario dan parameter yang telah ditentukan, berhasil membuat analisis terhadap kinerja *load balancing* menggunakan Algoritma Never Queue dibandingkan ketika diterapkan algoritma lain yaitu Algoritma Round Robin. Berikut adalah hasil dari analisis yang diperoleh dalam penelitian.
   a. Time Per *Request* pada *load balancing* yang menerapkan Algoritma Never Queue menghasilkan waktu respon yang akan lebih rendah jika *request* yang diterima lebih besar. Algoritma Never Queue menunjukkan stabilitas nilai time per *request* yang signifikan seiring dengan meningkatnya jumlah *request*. Sedangkan nilai Time Per *request* pada *load balancing* yang menerapkan Algoritma Round Robin menghasilkan waktu respon yang akan lebih tinggi jika *request* yang diterima lebih besar.
   b. *Request Per Second* pada *load balancing* yang menerapkan Algoritma Never Queue menghasilkan *Load balancing* yang memberikan pelayanan *request* yang terus meningkat di saat *request* yang diterima semakin besar. Sedangkan *Request Per Second* pada *load balancing* yang menerapkan

Algoritma Round Robin menghasilkan *load balancing* yang memberikan pelayanan *request* yang semakin rendah jika *request* yang diterima semakin besar.

c. Jumlah Koneksi pada *load balancing* yang menerapkan Algoritma Never Queue, ketiga *web* server menerima *request* dalam jumlah berbeda, pemilihan tersebut berdasarkan perhitungan Algoritma Shortest Expected Delay (SED). Hal ini yang mempengaruhi pendistribusian beban *request* menjadi tidak seimbang pada setiap server. Sedangkan jumlah koneksi pada *load balancing* yang menerapkan Algoritma Round Robin menghasilkan pembagian jumlah koneksi seimbang. Ketiga *web* server menerima *request* dalam jumlah yang relatif sama.

5. REFERENSI

[1] APJI. (2022). Profil Internet Indonesia 2022. Apji.or.Od, June. apji.or.id

[2] Riskiono, S. D., & Pasha, D. (2020). Analisis Metode *Load balancing* Dalam Meningkatkan Kinerja Website E-Learning. Jurnal Teknoinfo, 14(1), 22. https://doi.org/10.33365/jti.v14i1.466

[3] Rajguru, A., & Apte, S. (2012). A Comparative Performance Analysis of *Load balancing* Algorithms in Distributed System using Qualitative Parameters. International Journal of Recent Technology and …, 3, 175–179.

[4] Apriliansyah, F., Fitri, I., & Iskandar, A. (2020). Implementasi *Load balancing* Pada Web Server Menggunakan Nginx. Jurnal Teknologi Dan Manajemen Informatika,

[5] Bagus Ilham. (2021). Load balancing: Pengertian, Fungsi dan Cara Kerjanya pada Server. Niagahoster.Co.Id. https://www.niagahoster.co.id/blog/load-balancing-adalah/

[6] Nasser, H., & Witono, T. (2016). Analisis Algoritma Round Robin, Least Connection, Dan Ratio Pada *Load balancing* Menggunakan Opnet Modeler. Jurnal Informatika, 12(1), 25–32. https://doi.org/10.21460/inf.2016.121.455

[7] Alhaidari, F., & Balharith, T. Z. (2021). Enhanced round-robin algorithm in the cloud computing environment for optimal task scheduling. Computers, 10(5). https://doi.org/10.3390/computers10050063

[8] Sulaksono, D. H., & Giovanni, A. R. (2020). Implementasi Load balancingmenggunakan Antrian Round Robin Dengan Studi Kasuse-Shop. Jurnal Riset Inovasi BidangInformatikaDan Pendidikan Informatika,

[9] Febriantono, M. A. (2021). Round-Robin Scheduling. Binus.Ac.Id. https://binus.ac.id/malang/2021/11/round-robin-scheduling/

[10] Cassen, A. (2021). IPVS Scheduling Algorithms. https://keepalived-pqa.readthedocs.io/en/latest/scheduling_algorithms.html

[11] Alakeel, A. M. (2010). A Guide to Dynamic *Load balancing* in Distributed Computer Systems. IJCSNS International Journal of ComputerScience and Network Security, 10(6), 153–160. https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.456.5175&rep=rep1&type=pdf

[12] Xu, M., Tian, W., & Buyya, R. (2017). A Survey on *Load balancing* Algorithms for VM Placement in Cloud Computing. International Journal Of Computer Engineeringin Reseacrh Trends. https://doi.org/https://doi.org/10.1002/cpe.4123