ANALISIS KINERJA *LOAD BALANCING* MENGGUNAKAN ALGORITMA *DYNAMIC RATIO* PADA BEBAN TIGA *WEB SERVER*

ANALYSIS PERFORMANCE LOAD BALANCING USING DYNAMIC RATIO ALGORITHM ON THREE WEB SERVER LOADS

Ilham Reza Wijaya[1], Dr.Ir. Rendy Munadi., M.T.[2], Hafidudin., S.T., M.T.[3]

[1] Prodi S1 Teknik Telekomunikasi, [2]Fakultas Teknik Elektro, [3]Universitas Telkom

Jln. Telekomunikasi No.1 Terusan Buah Batu Bandung 40257 Indonesia

[1]ilhamrezawijaya@student.telkomuniversity.ac.id, [2]rendymunadi@telkomuniversity.ac.id [3]hafidudin@telkomuniversity.ac.id

ABSTRAK

Perkembangan aplikasi *website* berkembang dengan pesat. Hal ini membuat kebutuhan penggunaan akses suatu *website* meningkat. Dengan meningkatnya kebutuhan akan penggunaan suatu *website* maka membuat beban kerja *server* yang lebih pada suatu layanan *web server*, *server* menjadi tidak maksimal ketika permintaan dari pengguna meningkat, *server* akan terbebani karena harus melayani permintaan tersebut.

Metode yang dapat membagi trafik ke beberapa *server* sehingga tidak terjadi penumpukan beban pada sebuah *server* disebut *Load Balancing*. Perancangan *Load Balancing* pada penelitian ini bertujuan untuk meningkatkan kinerja pada sistem dan mendistribusikan beban yang tersedia pada *server*. Pertimbangan pembagian beban dapat dilakukan dengan menggunakan informasi perangkat pada *server* yang terdiri dari *CPU*, *memory* dan *disk*. Pada penelitian ini menggunakan algoritma *Dynamic Ratio* dan sebagai pembandingnya menggunakan algoritma *Round Robin*. Pengujian *Load Balancing* tersebut menggunakan *software* yang bernama *Httperf*. *Httperf* dapat menampilkan nilai parameter yang dibutuhkan seperti *Throughput*, *Response Time*, *Error* dan *CPU Utilization*.

Dari hasil pengujian menunjukan dengan menggunakan 3 *server* tidak terjadi *overload* pada *server* pada saat pengujian algoritma *Dynamic Ratio* maupun *Round Robin*. nilai rataan pada algoritma *Dynamic Ratio* sebesar 57.83 KB/s dan pada *Round Robin* sebesar 55.27 KB/s, nilai rataan *Response Time* pada agoritma *Dynamic Ratio* sebesar 2.64 detik dan *Round Robin* sebesar 2.67 detik, nilai *Error* pada algortima *Dynamic Ratio* sebesar 0.0% dan *Round Robin* sebesar 0.9% sedangkan nilai *CPU Utilization* pada algoritma *Dynamic Ratio* sebesar 93.5% dan algoritma *Round Robin* sebesar 93.0%. Nilai *Fairness Index* pada algoritma Dynamic Ratio tidak mencapai angka 1 sedangkan pada algoritma *Round Robin* dapat mencapai angka 1.

Kata kunci: *Load Balancing, Dynamic Ratio, Round Robin.*

ABSTRACT

The development of website applications is growing rapidly. This makes the need for access to a website increases. With the increasing need for the use of a website then create more server workload on a web server service, the server becomes not maximal when the demand from users increases, the server will be burdened because it must serve the request.

A method that can divide traffic to multiple servers so that no load buildup occurs on a server called Load Balancing. The design of Load Balancing in this study aims to improve performance on the system and distribute the load available on the server. Load sharing considerations can be done by using device information on a server consisting of CPU, memory and disk. In this study using the Dynamic Ratio algorithm and as a comparison using the Round Robin algorithm. Testing of Load Balancing uses software called Httperf. Httperf can display the required parameter values such as Throughput, Response Time, Error and CPU Utilization.

From the test results show that using 3 servers does not overload the server when testing the Dynamic Ratio and Round Robin algorithms. the average value of the Dynamic Ratio algorithm is 57.83 KB / s and Round Robin is 55.27 KB / s, the Response Time average value of the Dynamic Ratio algorithm is 2.64 seconds and Round Robin is 2.67 seconds, the value of Error in the Dynamic Ratio is 0.0% and Round Robin is 0.9% while the CPU Utilization value in the Dynamic Ratio algorithm is 93.5% and the Round Robin algorithm is 93.0%. The Fairness Index value in the Dynamic Ratio algorithm does not reach number 1 while in the Round Robin algorithm it can reach number 1.

Keywords: *Load Balancing, Dynamic Ratio, Round Robin.*

---

1. Pendahuluan

1.1 Latar Belakang Masalah

Perkembangan Perkembangan teknologi internet berkembang dengan pesat, salah satunya pada aplikasi *website*, aplikasi *website* bermacam-macam mulai dari *website ecommerce*, *website e-learning* dan masih banyak lagi. Karena hal itu, jumlah pengakses *website* pun juga ikut bertambah. Hal tersebut menyebabkan beban trafik yang cukup besar pada sebuah *server*. Saat jumlah pengakses cukup banyak dan mengakses bersamaan, itu menyebabkan *overload request* pada sebuah *server* yang menyebabkan kinerja *server* tidak maksimal disisi lain, pengakses menginginkan kecepetan akses yang maksimal.

Karena itu di butuhkan sebuah metode yang dapat mendistribusikan beban *server* secara merata, *Load Balancing* merupakan metode untuk membagi trafik ke beberapa *server* sehingga tidak terjadi penumpukan beban pada sebuah *server.* Pada Penelitian Sebelumnya membahas mengenai *Load Balancing* menggunakan algoritma *Rasio Dinamis* dan algoritma pembandingnya algoritma *Round Robin* menggunakan 2 *server* namun masih terjadi *overload*. Sehingga di dapatkan hasil *Load Balancing* dengan menggunakan algoritma *Rasio Dinamis* dapat memaksimalkan koneksi lebih besar dari 20.000 koneksi per detik. Sedangkan pada *Response Time* rataan waktu *Dynamic Ratio* lebih kecil dibandingkan dengan algoritma *Round Robin* [1].

Dalam penelitian ini untuk mendapatkan hasil yang maksimal penulis akan menganalisis kinerja *Load Balancing* dengan menggunakan 3 *server* untuk menganalisis kinerja algoritma *Dynamic Ratio* dan alogoritma *Round Robin.* Penggunaan algortima *Load Balancing* diharapkan dapat menghindari *overload* pada *server*. Parameter yang digunakan untuk mengalisis yaitu *Throughput, respon time, Error dan CPU Utilization*.

Dasar Teori

1.2 *Web Server*

Web server adalah software yang menjadi tulang belakang dari *world wide web (www). Web server* menunggu permintaan dari client yang menggunakan browser seperti Internet *Explorer, Mozilla Firefox*, dan program *browser* lainnya. Jika ada permintaan dari *browser*, maka web server akan memproses permintaan itu kemudian memberikan hasil prosesnya berupa data yang diinginkan kembali ke browser. Data ini mempunyai format yang standar, disebut dengan format SGML (*Standar General Markup Language*). Data yang berupa format ini kemudian akan ditampilkan oleh *browser* sesuai dengan kemampuan browser tersebut. bila data yang dikirim berupa gambar, browser yang hanya mampu menampilkan teks (misalnya lynx) tidak akan mampu menampilkan gambar tersebut, dan jika ada akan menampilkan alternatifnya saja. *Web server*, untuk berkomunikasi dengan *client*-nya (*web browser*) mempunyai protokol sendiri, yaitu HTTP (*hypertext transfer protocol*)

1.3 *Load Balancing*

Load Balancing adalah teknik yang mendistribusikan beban trafik secara seimbang agar trafik dapat berjalan optimal. Teknologi *Load balancing* di bangun pada jaringan yang menyediakan sebuah metode yang efektif untuk meningkatkan kemampuan dari perangkat jaringan seperti peningkatan *troughput*, menambah kemampuan data *processing* dan fleksibilitas *network*. Terutama menyelesaikan masalah

kongesti pada jaringan. Meningkatkan waktu tanggap pada *server* dan pemanfaatan *resource* server [3] *load balancing* dapat diimplementasikan dengan hardware maupun software.

1.4 Algoritma Dynamic Ratio

Algoritma *Dynamic Ratio* adalah algortima yang bobot rasionya didasarkan pada pemantauan server yang terus menerus dan oleh karena itu terus berubah. Algoritma *Dynamic Ratio* mendistribusikan koneksi berdasarkan berbagai aspek analisis kinerja *server* waktu nyata, seperti jumlah sambungan saat ini per node atau waktu respons node tercepat. Metode *Application Delivery Controller* ini jarang tersedia dalam *load balancer* sederhana. Perumusan penetuan rasio dapat dituliskan sebagai berikut.

$$r = X^{(c^{a-b}_{a})} + X^{(g^{e-f}_{e})} + X^{(j^{h-i}_{h})}$$

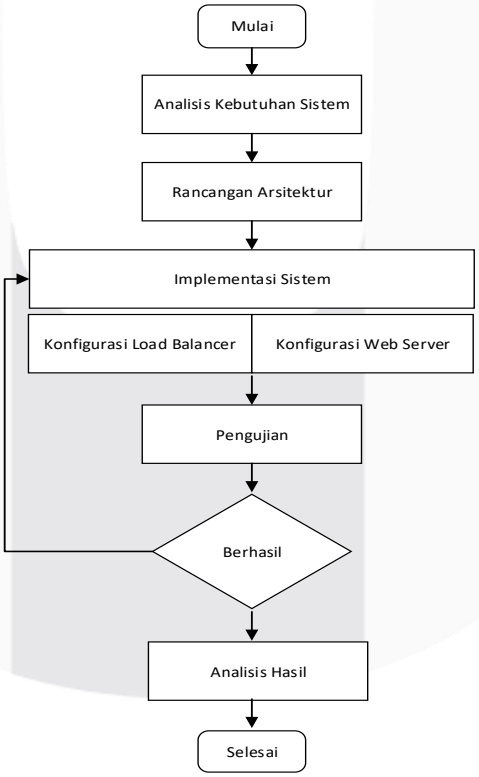
Keterangan:

X = number of nodes in Pool	*i = disk utilization*	*e = CPU Threshold*
a = memory threshold	*j = disk coefficient*	*f = CPU coefficient*
b = memory utilization	*h = disk Threshold*	*g = CPU coefficient*
c = memory coefficient		

2. Perancangan dan Implementasi Sistem

2.1 Perancangan Sistem

Berdasarkan diagram alir kerja sistem. Sistem ini menggunakan *F5 Network BIG-IP Load Balancer* sebagai media *load balancing*. Sistem ini akan bekerja sebagai sistem yang akan melakukan *load balancing* pada *client* yang memberikan *request* kepada server. Pada sistem ini menggunakan algoritma penjadwalan *Dynamic Ratio* yang akan di uji kinerjanya dan algoritma *Round Robin* sebagai algoritma pembandingnya.



Gambar 3.1 Diagram Alir Pengerjaan

2.2 Perangkat Keras dan Perangkat Lunak

Proses Proses analisis ini dikelompokan menjadi 2 bagian, yaitu analisis kebutuhan perangkat keras (*hardware*) dan analisis kebutuhan perangkat lunak (*software*) hasil dari analisis ini akan dituangkan dalam bentuk perancangan sistem. Perangkat keras yang digunakan pada penelitian ini adalah F5 Network BIG-IP dimana berfungsi sebagai sebuah perangkat yang sudah disiapkan untuk digunakan sebagai *Load Balancer*. Sedangkan perangkat lunak yang digunakan adalah sistem operasi Linux yaitu Linux Ubuntu sebagai *server* dan menggunakan Apache sebagai *web server*nya. Berikut adalah spesifikasi perangkat keras dan perangkat lunak yang digunakan pada penelitian tugas akhir ini adalah sebagai berikut.

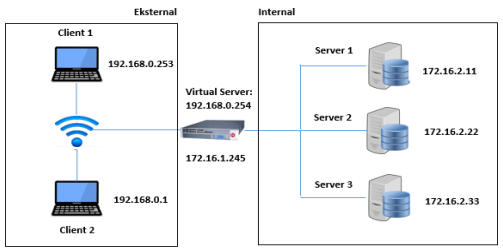
2.3 Perangkat Keras dan Perangkat Lunak

Perangkat-perangkat lunak yang digunakan untuk penelitian tugas akhir ini adalah sebagai berikut:

1. BIG-IP-13 sebagai load balancer
2. Linux Ubuntu sebagai sistem operasi pada server
3. Windows 10 64bit sebagai sistem operasi pada client.
4. Linux Ubuntu sebagai sistem operasi pada client
5. Httperf sebagai perangkat lunak yang digunakan untuk membangkitkan banyak request

2.4 Rancangan Arsitektur

Pada perancangan arsitektur penulis memilih mekanisme *Load Balancing* yang sesuai dengan situasi nyata untuk di implementasikan. Berikut rancangan simulasi *Load Balancing* dapat di lihat pada Gambar 3.2



Gambar 3.2 Rancangan Arsitektur

2.5 Implementasi Sistem

Pada tahap ini adalah tahap untuk melakukan konfigurasi Load Balancing pada F5 Network BIG-IP Load Balancer.

2.6 Pengujian

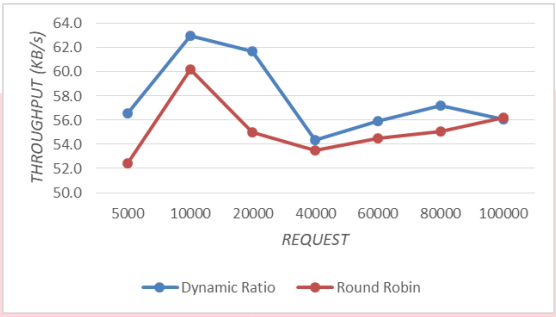
Tahap Pengujian merupakan tahap dilakukan pengujian terhadap agoritma Dynamic Ratio dengan menggunakan aplikasi httperf pada client 1 dan wireshark pada client 2. htperf adalah aplikasi yang berfungsi sebagai generator workload dan untuk mengambil data pengujian. Pengujian menggunakan httperf tidak hanya dilakukan satu kali pengujian, namun pada penelitian ini pengujian dilakukan 10 kali pengujian untuk mendapatkan hasil yang maksimal sedangkan wireshark adalah salah satu apikasi untuk memantau trafik jaringan pada saat melakukan pengujian. Parameter yang diujikan *adalah Throughput, Response Time, Error, CPU Utilization* dan *overload*. Pengujian dilakukan dengan menggunakan 5000 *request*, 10.000 *request*, 20.000 *request*, 40.000 *request*, 60.000 *request*, 80.000 *request* dan 100.000 *request* .

2.7 Analisis

Throughput

Parameter *Throughput* pada penelitian ini adalah jumlah *request* yang dapat direspon oleh *web server* pada satu waktu. Parameter ini dihitung dalam satuan KB/second. Semakin besar nilainya maka semakin baik

pula kinerja dari *Load Balancing* tersebut. Berikut adalah grafik hasil dari pengujian parameter *Throughput* pada algoritma *Dynamic Ratio* dan *Round Robin*.

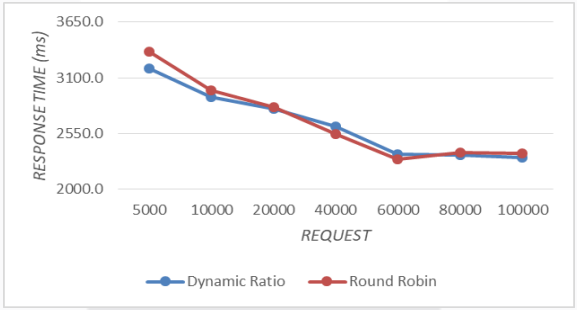


Gambar 4.3 Grafik rataan *Throughput Dynamic Ratio* dan *Round Robin*

Pada penelitian ini dapat dilihat grafik *Throughput* pada gambar 4.7 menunjukan nilai *Throughput* yang didapat dengan penggunaan kedua algoritma tersebut tidak terlalu signifikan perbedaannya grafik cenderung sama dan mengalami penurunan. Pada penelitian ini dengan menggunakan algoritma *Dynamic Ratio* memiliki rata-rata throughput yang lebih baik dibandingkan dengan menggunakan algoritma *Round Robin*. Pergerakan grafik maksimal pada saat 10000 *request* dimana pada saat 10000 *request* proses pembagian beban secara bergilirian dan berurutan dari satu *server* ke *server* lain lebih maksimal. nilai *Throughput* maksimal dicapai oleh keduanya pada saat pengujian 10000 *request* yaitu masing masing sebesar 63.0 KB/s dan 60.2 KBs setelah itu masing masing algoritma mengalami penurunan pada saat 20000 dan 40000 *request*. Hasil menunjukan nilai rataan *Throughput* pada algoritma *Dynamic Ratio* sebesar 57.83 KB/s dan nilai rataan *Throughput* pada algoritma *Round Robin* sebesar 55.27 KB/s. Hal tersebut dikarenakan pada algoritma *Dynamic Ratio* mendistribusikan *request* berdasarkan sambungan saat ini atau waktu *response node* tercepat dengan sehingga dapat mengurangi antrian yang membuat *server* dapat melayani *request* lebih banyak dan *Throughput* yang dihasilkan menjadi lebih besar.

Response Time

Parameter Response Time pada penelitian ini menggambarkan kecepatan suatu web server dapat melayani request dari sebuah client. Parameter ini dihitung dalam satuan milisecond. Semakin nilai parameternya kecil, maka sebuah web server dapat melayani request dari sebuah client semakin cepat.



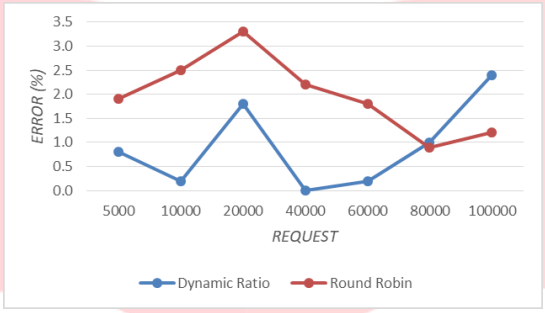
Gambar 4.4 Grafik rataan *Response Time Dynamic Ratio* dan *Round Robin*

Grafik rataan *Response Time* pada Gambar 4.8 menunjukan nilai rataan *Response Time* tidak terlalu jauh berbeda. Pada pengujian *Dynamic Ratio* dan *Round Robin* nilai signifikan terjadi pada awal percobaan dan cenderung menurun pada jumlah *request* berikutnya. Pada algoritma *Dynamic Ratio* didapatkan nilai rataan *Response Time* sebesar 2.64 detik dan pada algoritma *Round Robin* di dapatkan sebesar 2.67 detik. Dapat terlihat dari grafik, semakin sedikit *request* yang diberikan pada pengujian mengakibatkan nilai *Response Time* menjadi lebih besar seperti pada pengujian 5000 *request* didapatkan hasil *Response Time* masing masing sebesar 3.18 detik dan 3.35 detik. Nilai *Response Time* mengalami penurunan secara signifikan sampai pada pengujian 60000 *request* hal tersebut dikarenakan *server* mengalami lonjakan dan antrian secara mendadak sehingga mengakibatkan nilai *Response Time* menjadi tinggi dan cenderung sama setelah itu pada *server* telah mampu beradaptasi dengan banyaknya *request* yang diberikan kemudian masing masing

algoritma mengalami kestabilan. Pada pengujian masing masing algoritma juga diberikan batas waktu *timeout* selama lima detik hal ini membuat hasil pengujian *response time* cenderung tidak jauh berbeda.

Error

Parameter *Error* menggambarkan tentang banyaknya *Error* yang terjadi pada suatu *server* menanggapi *request* dari *client*. Semakin kecil nilai *Error*, maka semakin baik kinerja sebuah *server* dalam menanggapi *request* dari *client*.

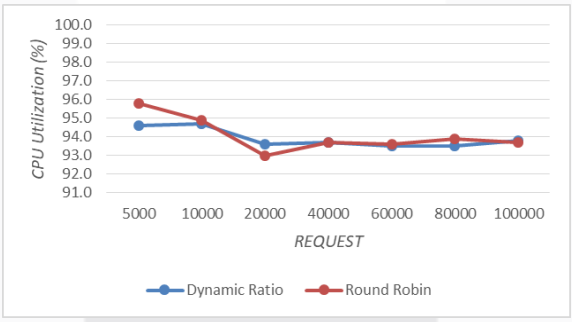


Gambar 4.5 Grafik rataan *Error Dynamic Ratio* dan *Round Robin*

Pada gambar 4.9 menunjukkan grafik rataan perbandingan nilai *Error* antara algoritma *Dynamic Ratio* dan *Round Robin*. pada grafik diatas dapat dilihat hasil *Error* algoritma *Round Robin* lebih banyak dibandingkan algoritma *Dynamic Ratio*. Pada saat 20000 *request* kedua algoritma sama sama mengalami kenaikan dan cenderung menurun namun pada algoritma *Dynamic Ratio* dan *Round Robin* pada saat 20000 *request* dan 100000 *request* terjadi *Error* yang cukup signifikan. nilai *Error* terendah ada pada algoritma *Dynamic Ratio* pada saat pengujian 40000 *request* yaitu sebesar 0.0% dengan kata lain pada saat pengujian 40000 request algoritma *Dynamic Ratio* tidak mengalami *Error* pada *server* sedangkan nilai *Error* terbesar ada pada algoritma *Round Robin* saat pengujian 20000 *request* yaitu sebesar 3.3%. Hal tersebut dikarenakan pada saat pendistribusian *request Round Robin* terjadi penumpukan yang membuat *request* dari *client* yang ditolak atau tidak mampu terjawab oleh *server* lebih banyak sedangkan pada algoritma *Dynamic Ratio* mendistribusikan *request* tidak melihat urutan *server*

CPU Utilization

Parameter *CPU Utilization* digunakan untuk mengetahui sumber daya yang diperlukan dalam komputerisasi. *CPU Utilization* dihitung dalam bentuk satuan persen (%). Sehingga semakin kecil nilai persentasenya, maka akan semakin sedikit *source* yang digunakan dalam proses *Load Balancing*.



Gambar 4.6 Grafik rataan *CPU Utilization Dynamic Ratio* dan *Round Robin*

Pada gambar 4.10 dapat dilihat hasil dari besarnya penggunaan daya CPU dengan menggunakan algoritma *Dynamic Ratio* dan *Round Robin*. Pada pengujian *CPU Utilization* dapat dilihat perbedaan grafik ada pada pengujian 5000 *request* hal tersebut dikarenakan pada *server* belum dapat beradaptasi dengan antrian *request* yang dikirim *client* sedangkan *server* baru dapat menangani *request* pada saat 10000 *request* sampai dengan 100000 *request*. Pada saat 20000 *request* dapat terlihat perbedaan daya yang digunakan. Pada 5000 *request* daya yang digunakan algoritma *Round Robin* lebih besar dibandingkan algoritma *Dynamic Ratio* yaitu sebesar 95.8% sedangkan pada *Dynamic Ratio* sebesar 94.7% kemudian server mengalami penurunan nilai *CPU Utilization* sampai 20000 *request* hal tersebut dikarenakan pada algoritma *Round Robin* mendistribukan *request* tidak melihat kondisi status *server* sedangkan pada algoritma *Dynamic Ratio*

mendistribukan *request* berdasarkan status *server* terendah yang membuat penggunaan daya lebih stabil dibandingkan algoritma *Round Robin*.

Overload

Pada Tugas akhir ini dilakukan pengujian dengan menggunakan 3 server hal ini dimaksudkan untuk mengurangi beban pada server. Dari hasil pengujian didapatkan ketiga server pada load balancer tidak terjadi overload, berbeda dengan hasil penelitian sebelumnya yang menggunakan 2 server terjadi overload pada algoritma Round Robin, pada load balancer kondisi real server dalam sistem dapat diketahui secara otomatis, apabila pada load balancer terdapat real server yang mati makan akan di hapus dari daftar real server dan jika real server sudah aktif kembali maka akan dimasukan kembali ke dalam sistem.

Fairness Index

Pengukuran *Fairness Index* bertujuan untuk mengetahui seberapa *fair* algoritma *Dynamic Ratio* dan *Round Robin* dapat mendistribusikan sumber daya yang adil dan merata pada setiap *server cluster* yang ada pada *load balancer*.

Tabel 4.7 Hasil Fairness Index pada Algoritma Dynamic Ratio dan Round Robin

Jumlah *Request*	Nilai *Fairness Index*	
	Dynamic Ratio	*Round Robin*
5000	0.99956012	0.99999992
10000	0.99977551	0.99999998
20000	0.99984263	0.99999995
40000	0.99997489	0.99999998
60000	0.99996422	1
80000	0.99976318	0.99999997
100000	0.99925052	0.99999998

Berdasarkan pada Tabel 4.7 hasil pengukuran parameter Fairness Index seperti yang ditunjukan dapat terlihat bahwa nilai Fairness yang dihasilkan oleh algoritma Round Robin hampir mendekati angka 1 pada jumlah request 5000, 10000, 20000, 40000, 80000 dan 100000 sedangkan pada jumlah request 60000 mendapatkan angka Fairness Index yaitu 1. Hasil pengukuran secara keseluruhan menunjukkan pada algoritma Round Robin memiliki nilai Fairness Index yang lebih unggul dibandingkan algoritma Dynamic Ratio, artinya load balancer dengan menggunakan algoritma Round Robin lebih adil dalam mendistribusikan beban request. Hal ini dikarenakan proses pembagian beban secara berurutan dari satu server ke server lainnya sehingga pembagian pada algoritma Round Robin lebih merata dibandingkan algoritma Dynamic Ratio.

5. Kesimpulan

Berdasarkan hasil dari penelitian ini dapat ditarik kesimpulan sebagai berikut :

1. *F5 Network BIG-IP Load Balancer* dapat menjalankan algoritma *Dynamic Ratio* dan *Round Robin.*
2. CPU, *memory* dan *disk* adalah parameter untuk mempertimbangkan pembagian rasio pada algoritma *Dynamic Ratio.*
3. nilai rataan *Throughput* pada algoritma *Dynamic Ratio* lebih besar dibandingkan dengan algoritma *Round Robin*.
4. Hasil pengujian parameter *Response Time* algoritma *Dynamic Ratio* lebih cepat dibandingkan algoritma *Round Robin*.
5. Hasil pengujian parameter *Error* algoritma *Dynamic Ratio* lebih sedikit mengalami *Error* dibandingkan algoritma *Round Robin.*
6. Hasil pengujian parameter *CPU Utilization* algortima *Round Robin* lebih sedikit menggunakan daya pada saat melakukan komputerisasi atau proses *load balancing*
7. Dengan menggunakan 3 *server* tidak terjadi *overload* karena *High Availabilty* 3 *server* lebih baik dibandingkan penelitian sebelumnya yang menggunakan 2 *server*
8. Dengan Nilai *Fairness Index* pada algoritma *Round Robin* lebih unggul dibandingkan algoritma *Dynamic Ratio* dikarenakan pada algoritma *Round Robin* sanggup mencapai angka 1 pada 60000 *request*.

DAFTAR PUSTAKA

[1] Aris Munandar. Laporan Tugas Akhir Dengan Judul "Analisis Kinerja Penyeimbang Beban Server Web dengan Algoritme Rasio Dinamis", Institut Pertanian Bogor.2014

[2] Gian Rahayu. Laporan Tugas Akhir Dengan Judul "Rancang Bangun Web Server untuk Pemantauan Budidaya Udang Vannamei menggunakan Teknologi IOT", Telkom University.2017.

[3] Shang, z., Chen, w., Qiang, m., & Bin, W. (2011). Design and implementation of server cluster dynamic load balancing based on openflow. 2.

[4] Faris Putra Perdana. Laporan Tugas Akhir Dengan Judul "Analisis Performansi Load Balancing dengan Algoritma Weighted Round Robin pada Software Define Network (SDN))", Telkom University.2017

[5] Mohammad Rizky Pratama. Laporan Tugas Akhir Dengan Judul "Analisis Performansi Load Balancing dengan Algortima Round Robin dan Least Connection pada sebuah Web Server", Telkom University.2017

[6] K6406: Overview of Dynamic Ratio load balancing. 05 Februari 2018. https://support.f5.com/csp/article/K6406

[7] Maya Rosalia. Laporan Tugas Akhir Dengan Judul "Implementasi High Availability Server Menggunakan Metode Load Balancing dan Failover pada Virtual Web Server Cluster", Telkom University.2016

[8] Satria Geusan. Laporan Tugas Akhir Dengan Judul "Analisa Kinerja Load Balancing Menggunakan Linux Virtual Server pada Layanan HTTP", Telkom University.2016

[9] Abdul Haris. Laporan Tugas Akhir Dengan Judul "Komparasi Algoritma Penjadwalan Pada Layanan Terdistribusi Load Balancing LVS VIA NAT", Insititut Teknologi Sepuluh Nopember.2011