Coding : Jurnal Komputer dan Aplikasi
Volume 09, No. 01 (2021), hal 131-141 ISSN : 2338-493X

ANALISA KINERJA *LOAD BALANCING* MENGGUNAKAN METODE *ROUND ROBIN* DAN *WEIGHTED ROUND ROBIN*

Ricky Oktariyadi[1], Ikhwan Ruslianto[2] , Syamsul Bahri[3]
[1,2,3] Jurusan Rekayasa Sistem Komputer, Fakultas MIPA Universitas Tanjungpura

Jl. Prof. Dr. H. Hadari Nawawi, Pontianak
Telp/Fax.: (0561) 577963
e-mail: [1]rickyo@student.untan.ac.id, [2]ikhwanruslianto@siskom.untan.ac.id,
[3]syamsul.bahri@siskom.untan.ac.id

ABSTRAK

Peningkatan jumlah pengunjung web dapat menyebabkan kelebihan beban pada server. Kelebihan beban ini dapat menyebabkan server berhenti bekerja. Hal yang dapat dilakukan untuk mengatasi kelebihan beban ini salah satunya adalah dengan menerapkan load balancer pada server. Metode load balancer adalah metode yang digunakan untuk mendistribusikan request atau beban kerja kepada beberapa server atau komputer untuk meminimalkan waktu respon dan menghindari kelebihan beban. Dalam penggunaannya, load balancer menggunakan algoritma penjadwalan dalam proses pendistribusian beban kerja atau request kepada beberapa komputer atau server. Penelitian ini bertujuan untuk membandingkan nilai response time dari load balancer dengan metode round robin dengan load balancer dengan metode weighted round robin. Berdasarkan hasil pengujian 10 sampel pengujian, load balancer dengan metode round robin akan membagikan beban request secara merata pada web server sedangkan load balancer dengan metode weighted round robin akan membagikan beban request sesuai dengan beban yang telah diberikan pada web server. Penelitian ini juga menyimpulkan semakin besar jumlah request yang dikirim maka nilai response time juga akan semakin besar. Perbandingan nilai response time total load balancing metode weighted round robin dengan beban 1:2 dan load balancing metode round robin yang mempunyai spesifikasi server yang sama akan menghasilkan selisih persentase response time total antara 13% sampai dengan 46%.

Kata Kunci : *Load balancer, Round Robin, Weighted Round Robin, Response Time*

1. PENDAHULUAN

Berdasarkan hasil data BPS (Badan Pusat Statistik) pada tahun 2020, pengguna internet di Indonesia mengalami peningkatan jumlah sebesar 25,5 juta pengguna. Peningkatan jumlah pengguna internet menyebabkan terjadinya peningkatan jumlah pengunjung situs web. Peningkatan jumlah pengujung situs web dapat menyebabkan kelebihan beban pada server. Untuk mengatasi kelebihan beban ini, cara yang digunakan salah satunya adalah dengan menggunakan *load balancer* pada server. *Load balancer* bekerja dengan cara mendistribusikan *request* atau beban kerja kepada beberapa komputer atau server lain. Dalam penerapannya, *load balancer* menggunakan metode penjadwalan dalam proses pendistribusian beban kerja atau *request* kepada web server lain. Metode penjadwalan yang paling sering digunakan salah satunya adalah *round robin* [1].

Penelitian yang dilakukan oleh Supramana berjudul ”Implementasi *Load Balancer* Pada Web Server Dengan Mengunakan Apache” menyimpulkan bahwa sebagai *load balancer* Apache membuat beban yang berurutan pada setiap web server serta sama tanpa memiliki beban lebih, dan memiliki waktu tanggap lebih tinggi pada web server [1]. Penelitian yang dilakukan oleh Riskiono berjudul “Analisa Metode *Load Balancing* Dalam Meningkatkan Kinerja Website *E-Learning*” menyimpulkan bahwa *load balancing* yang diimplementasikan memiliki nilai *response time* 36,4 ms lebih kecil dibandingkan dengan server tunggal yang memiliki nilai *response time* 51,1 ms pada uji koneksi 500/10 sec. Penelitian yang dilakukan oleh Riskiono juga menyimpulkan bahwa penerapan *load balancing* lebih baik dibandingkan dengan server tunggal dari segi nilai *response time* jika untuk setiap rentang pengujiannya [2].

Penelitian yang dilakukan oleh Rahmatulloh berjudul "Implementasi *Load Balancing* Web Server Menggunakan HAProxy Dan Sinkronisasi *File* Pada Sistem Informasi Akademik Universitas Siliwangi" menyimpulkan bahwa *load balancing* bekerja dengan baik ketika *request* yang datang dari *client* berhasil didistribusikan *load balancer* secara merata kepada setiap node cluster sehingga *overload* tidak terjadi pada server dan web server bisa memiliki kemampuan melayani 10.000 *request* dengan tidak mengalami *error request*. Selain itu, penerapan sinkronisasi *file* telah bekerja dengan baik dimana semua file yang diunggah pada node 1 akan disinkron kepada setiap node 2 dan kepada node 3 pada *cluster*, begitu juga sebaliknya karena sinkronisasi *file* ini bersifat dua arah [3]. Penelitian yang dilakukan oleh Kuncor berjudul "Implementasi Health Checks Monitoring Pada *Load Balancer* Web Server Berbasis Android" menyimpulkan bahwa penggunaan *load balancer* memiliki pengaruh bagi server dari segi peningkatan kecepatan dan penanganan *request* sebesar 67% kepada pengujian pertama dan 72% kepada pengujian kedua. Penelitian yang dilakukan oleh Kuncoro juga menyimpulkan bahwa sistem *health check* berhasil memperoleh data sumber daya dari server dan berhasil mencegah pengiriman *request* kepada server yang sedang bermasalah [4].

Berdasarkan kesimpulan penelitian yang telah dilakukan oleh Supramana, Riskiono, Rahmatulloh, dan Kuncoro penggunaan *load balancing* masih menggunakan metode *round robin* dimana metode ini akan membagikan beban *request* yang masuk kepada *load balancer* secara merata kepada server lain.

Berdasarkan penelitan yang telah dijelaskan sebelumnya, *load balancer* pada penelitian tersebut masih menggunakan metode *round robin* sebagai metode pembagian beban *load balancer*. Metode ini memiliki kekurangan yaitu metode ini tidak menghiraukan kondisi server saat dikirimi beban *request* oleh *load balancer*. Penelitian sebelumnya juga belum membandingkan metode pembagian beban lain yang digunakan pada load balancer. Karena itu dilakukanlah penelitian yang berjudul "Analisa Kinerja *Load Balancing* Menggunakan Metode *Round Robin* Dan *Weighted Round Robin*" sebagai solusi metode pembagian beban lain pada *load balancer* serta menganalisa kinerja *load balancing* metode *round robin* dan metode *weighted round robin.*

2. LANDASAN TEORI

2.1. *Load Balancing*

Load balancing adalah metodologi jaringan pada komputer yang bekerja dengan cara mendistribusikan *request* atau beban kerja kepada beberapa server atau *cluster server* untuk meminimalkan waktu respon dari server, dan menghindari kelebihan beban pada server. *Cluster server* adalah penggabungan dari beberapa perangkat komputer atau server yang saling terhubung satu sama lain dan saling bekerja sama sehingga dapat dilihat sebagai 1 sistem yang sama dalam banyak aspek, dan *cluster server* digunakan untuk meningkatkan kinerja dan ketersediaan dari sebuah komputer. Waktu respon adalah waktu yang diperlukan server untuk memproses request yang masuk [5].

Load balancer juga dapat melakukan berbagai macam fungsi seperti *traffic engineering*, penyeimbang beban, serta fungsi pemindahan jalur trafik. Dalam penerapannya, load balancer memiliki berbagai macam algoritma penjadwalan yang dapat digunakan untuk mendistribusikan beban kepada setiap server yang berada didalam sekumpulan server [5].

2.1.1 *Load Balancing* Dengan Algoritma Penjadwalan *Round Robin*

Load balancing dengan algoritma penjadwalan *round robin* adalah *load balancing* yang bekerja dengan cara mendistribusikan beban kerja atau *request* yang diterima sama rata kepada seluruh real server tanpa memperdulikan kapasitas server tersebut atau pun beban permintaan layanan yang dikirimkan. Jika ada 2 real server (A,B), maka beban kerja atau *request* pertama akan diberikan *load balancer* kepada server A, beban kerja atau *request* kedua akan diberikan kepada server B, dan beban kerja atau *request* ketiga akan diberikan kembali ke server A. Mekanisme ini dapat dilakukan ketika seluruh real server yang dipakai menggunakan spesifikasi komputer yang sama. Konsep dasar yang digunakan algoritma ini adalah *timesharing.* Pada dasarnya algoritma ini bekerja dengan cara *First Come First Serve* (FCFS), hanya saja metode ini bersifat preemptive, dimana setiap proses yang terjadi mendapatkan waktu CPU atau waktu kuantum (*quantum time*) untuk membatasi waktu proses.Waktu ini

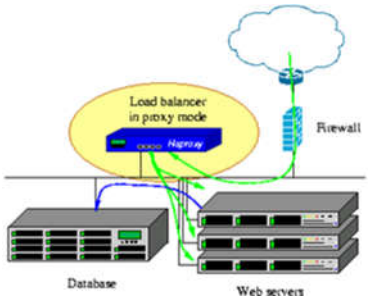
biasanya bernilai sebesar 1-100 milidetik. Setelah waktu tersebut habis, proses kemudian ditunda dan ditambahkan kepada *ready queue*. Algoritma *round robin* merupakan algoritma yang paling sederhana dan banyak digunakan oleh perangkat *load balancing*. Algoritma round robin membagi beban web server secara bergiliran dan berurutan antara web server yang satu dengan web server lainnya sehingga penjadwalan akan membentuk sebuah putaran [6].

2.1.2 *Load Balancing* Dengan Algoritma Penjadwalan *Weighted Round Robin*

Load balancing dengan algoritma penjadwalan *weighted round robin* adalah algoritma yang memiliki cara kerja yang sama dengan algoritma *round robin* akan tetapi sebuah kondisi baru diberikan kepada server yang akan dikirimkan *request* atau beban kerja dari *load balancer*. Kondisi baru ini ditambahan dengan mempertimbangkan kemampuan sumber daya dari server yang akan digunakan dan memberikan jumlah tugas yang lebih besar kepada server yang mempunyai kapasitas yang lebih tinggi [7]. Setiap antrian beban kerja atau *request* yang diberikan kepada server sudah ditentukan terlebih dahulu, yang mana akan mendetailkan jumlah beban kerja atau *request* yang akan ditransmisikan dalam 1 kali proses pendistribusian [8]. Perbandingan beban kerja atau *request* sebesar 1:2 digunakaan saat *cluster server* yang digunakan hanya 2. Ini berarti server 1 akan mendapatkan beban kerja atau *request* sebesar 1 dan server 2 akan mendapatkan beban kerja atau *request* sebesar 2. Ketika *load balancer* mendistribusikan 3 beban kerja atau *request*, maka beban kerja atau *request* pertama akan dikirimkan kepada server 1 dan beban kerja atau *request* kedua dan ketiga akan dikirimkan kepada server 2 [9].

2.2. HAProxy

HAProxy adalah *load balancer* proxy server yang menawarkan *load balancing* yang tersedia untuk proxy *Transmission Control Protocol* (TCP) dan Hypertext Transfer Protocol (HTTP) berbasis aplikasi. HAProxy sendiri bekerja dengan berintegrasi dengan arsitektur yang sudah ada sehingga lebih mudah dan tidak beresiko, sementara masih menawarkan kemungkinan untuk tidak menampilkan web server yang rentan terhadap internet seperti Gambar 1 [10].



Gambar 1 Arsitektur HAProxy
sumber: (haproxy.org)

2.3. Web Server

Web server adalah sebuah perangkat lunak yang melayani permintaan HTTP yang dikirimkan dari web browser serta mengirimkan kode-kode dinamis kepada server aplikasi. Web server bertugas untuk mengirimkan transfer berkas permintaan pengguna melalui protokol komunikasi yang telah ditentukan sedemikian rupa. Halaman web yang diminta dari web browser terdiri dari berkas gambar, teks, dan lain-lain [11].

2.4. HTML

Hyper Text Markup Language (HTML) adalah bahasa *markup* yang digunakan untuk membuat halaman web. HTML menyusun halaman web menjadi sedemikian rupa sesuai halaman web yang kita lihat melalui browser. HTML bukanlah bahasa pemrograman. HTML adalah sebuah bahasa *markup* yang berisikan perintah-perintah dengan format yang terstruktur untuk menampilkan tampilan sesuai dengan format yang digunakan [12].

2.5. PHP

Hypertext Preprocessor (PHP) diciptakan oleh Ramus Lerdorf, seorang pemogram C yang andal. PHP merupakan bahasa *server-side scripting* yang bersatu dengan HTML untuk membuat sebuah halaman web yang dinamis. Maksud daripada *server-side scripting* adalah syntax dan perintah-perintah yang digunakan akan sepenuhnya dijalankan pada server tetapi disertakan kedalam dokumen HTML.

Kelebihan-kelebihan dari PHP di antaranya adalah PHP secara dasar dapat mengerjakan semua perkerjaan yang dikerjakan oleh program *Computer Generated Imagery* (CGI), seperti menghasilkan isi halaman web yang dinamik, pekerjaan mendapatkan data dari form, dan menerima *cookies*. Kemampuan yang paling

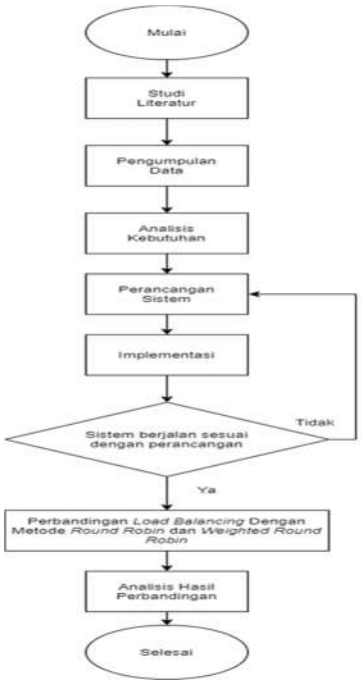
dapat diandalkan dan paling signifikan adalah dukungan kepada berbagai banyak basis data [13].

3. METODOLOGI PENELITIAN

Metode penelitian merupakan rencana penelitian untuk menerapkan sistem yang dibuat. Penelitian dilakukan dengan 7 tahap yaitu studi literatur, metode pengumpulan data, analisis kebutuhan, perancangan sistem, implementasi, pengujian, dan analisis hasil pengujian yang ditunjukkan pada Gambar 2.

3.1 Studi Literatur

Pada tahap ini dilakukan penelusuran yang berkaitan dengan penelitian tugas akhir. Studi literatur dilakukan sebagai pengumpulan bahan-bahan referensi yang akan digunakan. Literatur yang digunakan berupa buku-buku, jurnal ilmiah penelitian sebelumnya, serta berbagai data-data yang bisa digunakan untuk mendukung penyelesaian penelitian tugas akhir



Gambar 2 *Flowchart* Penelitian

3.2 Metode Pengumpulan data

Metode pengumpulan data yang dilakukan dalam penelitian adalah metode observasi. Metode observasi adalah metode pengamatan secara langsung kegiatan yang sedang dilakukan pada penelitian ini. Pengumpulan data dilakukan dengan melakukan simulasi pengujian kepada *load balancing* server. Simulasi pengujian dilakukan dengan mengirimkan sampel *request* kepada alamat IP load balancer. Data yang dikumpulkan merupakan nilai *response time* dari masing-masing metode *load balancing* yang telah disimulasikan. Ketika melakukan observasi juga bisa melakukan validasi dari informasi yang didapat.

3.3 Analisis Kebutuhan

Analisis Kebutuhan adalah sebuah proses untuk mendapatkan informasi dari spesifikasi perangkat lunak serta perangkat keras terkait dengan penelitian ini tentang analisa kinerja *load balancing* metode *round robin* dan *weighted round robin*.

3.3.1 Kebutuhan Perangkat Keras

Perangkat keras yang digunakan pada penelitian ini dapat dilihat pada Tabel 1.

Tabel 1 Kebutuhan Perangkat Keras

No	Komponen	Keterangan	Spesifikasi
1	Komputer	Mengakses dan konfigurasi server, konfigurasi PHP	Prosesor: AMD A6-6310 1,8 GHz, RAM: 8 GB, SSD: 128 GB
2	*Virtual private server* (VPS)	Menjadi server *load balancer* dan web server.	1vCPU

3.3.2 Kebutuhan Perangkat Lunak

Perangkat lunak yang digunakan dapat dilihat pada Tabel 2.

Tabel 2 Kebutuhan Perangkat Lunak

No	Komponen	Keterangan	Spesifikasi
1	Windows 10	Sistem Operasi	Ver 2004
2	Ubuntu Server	Sistem Operasi VPS	Ver 18.04
3	Apache	Web Server	Ver 2.4.29
4	HAProxy	Load Balancer	Ver 1.8.8
5	Putty	Akses VPS lewat Komputer	Ver 0.74
6	Draw.io	Menggambar diagram	Ver 13.9.9
7	Sublime	Mengedit kode program	Ver 3.2.2
8	Google Chrome	Web Browser	Ver 87.0.4280.88

Tabel 2 Kebutuhan Perangkat Lunak

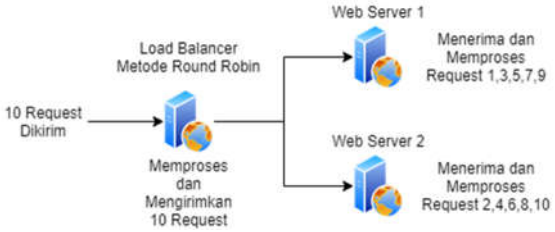
9	Apache Benchmark	Mengirimkan *request* kepada server *load balancer*	Ver 2.4.29

3.4 Perancangan Sistem

Perancangan sistem dibuat dan dirancang berdasarkan penelitian yang akan dilakukan. Pada penelitian ini, tahap perancangan terbagi menjadi 3 buah tahapan, perancangan sistem *load balancer metode round robin*, perancangan sistem *load balancer* metode *weighted round robin* dan perancangan antarmuka. Perancangan sistem ini bertujuan agar sistem yang akan dibuat lebih terarah juga sebagai panduan dalam implementasi sistem ke dalam kode program.

3.4.1 Perancangan Sistem *Load Balancer* Metode *Round Robin*

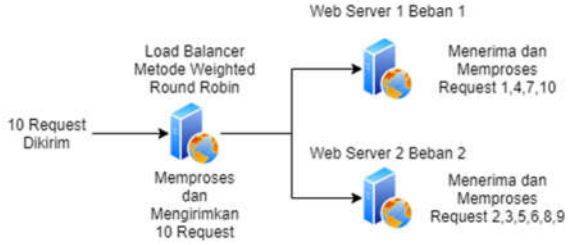
Perancangan sistem *load balancer* metode *round robin* dilakukan untuk membuat rancangan sistem yang akan digunakan sebagai panduan dalam pembuatan sistem. Tahap perancangan ini didasarkan pada daftar analisis kebutuhan. Pada tahap ini, rancangan sistem digambarkan dengan rancangan alur kerja sistem.



Gambar 2 Rancangan Metode *Round Robin*

3.4.2 Perancangan Sistem *Load Balancer* Metode *Weighted Round Robin*

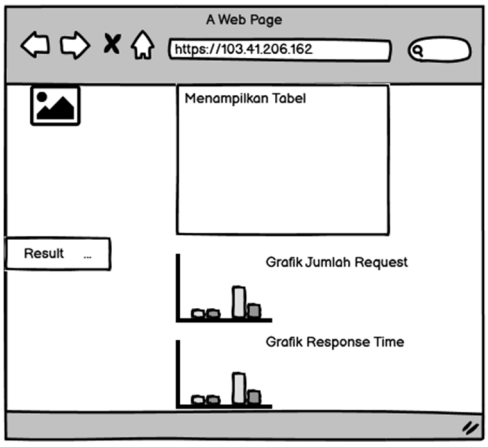
Perancangan sistem *load balancer* metode *weighted round robin* dilakukan untuk membuat rancangan sistem yang akan digunakan sebagai panduan dalam pembuatan sistem. Tahap perancangan ini didasarkan pada daftar analisis kebutuhan. Pada tahap ini, rancangan sistem digambarkan dengan rancangan alur kerja sistem.



Gambar 3 Rancangan Metode *Weighted Round Robin*

3.4.3 Perancangan Antarmuka

Perancangan antarmuka merupakan tahapan yang dilakukan untuk merancangan antarmuka sistem. Perancangan ini dilakukan untuk memberikan kelancaran ketika mengakses informasi yang dibutuhkan sehingga mengurangi tingkat kebingungan dalam menggunakan sistem. Tahapan pada proses perancangan antarmuka pada sistem analisa yang akan dibuat yaitu proses perancangan antarmuka tampilan *result* pada halaman web. Tampilan result berisi tabel hasil *benchmark*, grafik jumlah request, dan grafik response time.



Gambar 4 Rancangan Antarmuka

3.5 Implementasi

Implementasi dari penelitian ini terbagi menjadi 3 proses utama. Proses ini terdiri atas proses implementasi sistem *load balancing* metode *round robin* pada server, proses implementasi sistem *load balancing* metode *weighted round robin* pada server, dan proses menampilkan antarmuka sistem. Proses implementasi sistem *load balancing* metode *round robin* dan *weighted round robin* ditampilkan dalam bentuk *screenshoot* gambar kode program yang telah berhasil dikonfigurasi pada HAProxy. Proses menampilkan antarmuka sistem ditampilkan dalam bentuk *screenshoot*

gambar tampilan result dalam halaman web. Langkah-langkah untuk proses implementasi dan menampilkan antarmuka dipisahkan ke dalam beberapa proses.

3.6 Pengujian

Pengujian dilakukan dengan mengirimkan sampel pengujian kepada server *load balancing*. Sampel pengujian yang digunakan sebanyak 10 sampel pengujian. Proses pengujian dilakukan dengan cara mengirimkan *request* kepada server *load balancing*. *Request* dikirimkan melalui *tools* Apache Benchmark kepada server *load balancing*. Hasil *response time* dari pengujian ini kemudian akan diambil untuk dianalisa perbandingan nilainya.

3.7 Analisa Hasil Pengujian

Analisa hasil pengujian dilakukan setelah hasil *response time* dari sampel pengujian dikumpulkan. Hasil *response time* yang telah dikumpulkan ini kemudian dianalisa dan dibandingkan antara hasil *response time* metode *round robin* dan metode *weighted round robin*.

4 HASIL DAN PEMBAHASAN

Hasil pengujian ditampilkan dalam bentuk tangkapan layar dari web. Hasil tangkapan layar ditunjukan pada Gambar 5, Gambar 6, Gambar 7, Gambar 8, Gambar 9, dan Gambar 10.

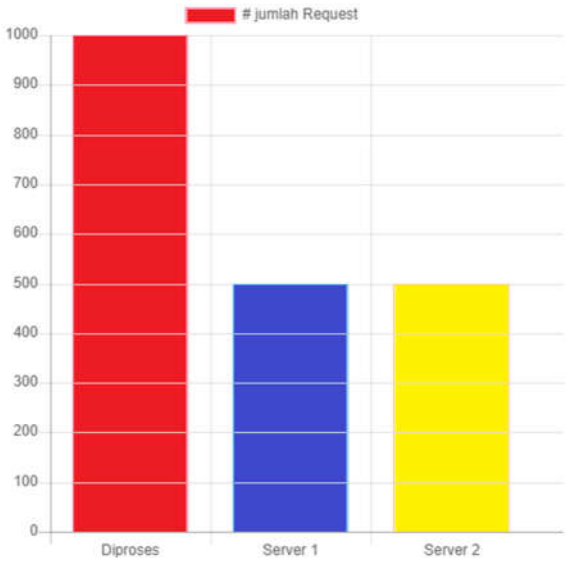
Result

Ini adalah tabel hasil benchmark.

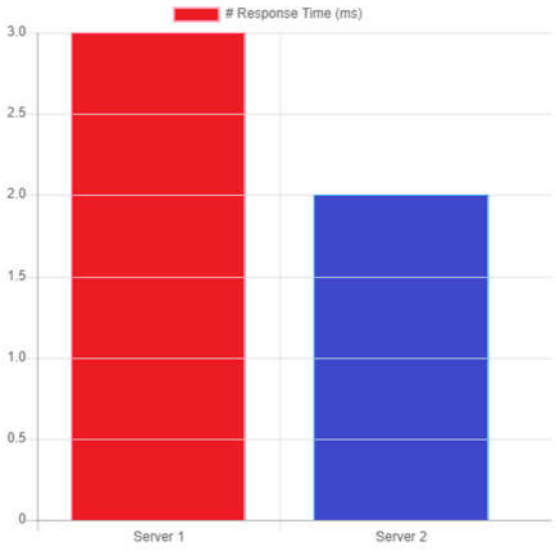
Total Request Yang Diproses Load Balancer : 1000 request

Nama Server	Request Masuk	Response Time (ms)
Web Server 1	500 request	3
Web Server 2	500 request	2

Gambar 5 Tabel Hasil *Benchmark Round Robin*



Gambar 6 Grafik Jumlah *Request Round Robin*



Gambar 7 Grafik *Response Time Round Robin*

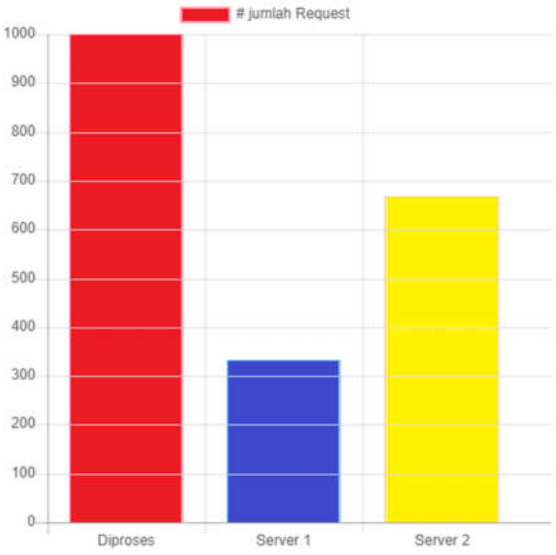
Result

Ini adalah tabel hasil benchmark.

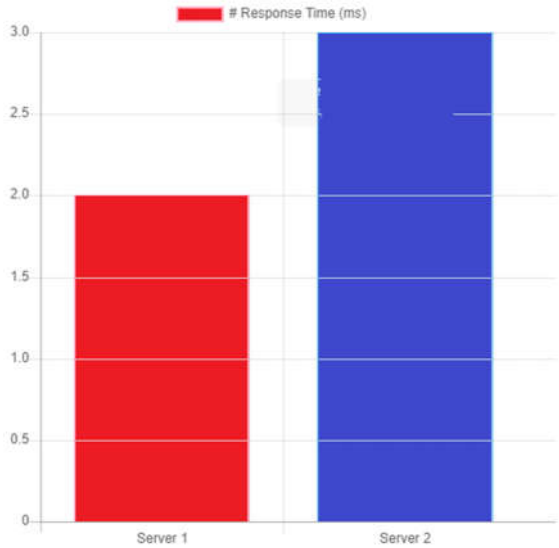
Total Request Yang Diproses Load Balancer : 1000 request

Nama Server	Request Masuk	Response Time (ms)
Web Server 1	333 request	2
Web Server 2	667 request	3

Gambar 8 Tabel Hasil *Benchmark Weighted Round Robin*



Gambar 9 Grafik Jumlah *Request Weighted Round Robin*



Gambar 10 Grafik *Response Time Weighted Round Robin*

Gambar 5 merupakan tabel hasil *benchmark* yang menampilkan hasil pengujian terhadap sistem *load balancing* metode *round robin*. Tabel ini berisikan jumlah *request* yang diproses *load balancer*, jumlah *request* yang dikirim pada web server 1, jumlah *request* yang dikirim pada web server 2, nilai *response time* web server 1, dan nilai *response time* web server 2. Gambar 6 merupakan grafik batang jumlah *request*. Nilai yang terdapat pada grafik ini sama dengan nilai yang terdapat pada Gambar 5. Gambar 7 merupakan grafik batang nilai *response time*. Nilai yang terdapat pada grafik ini sama dengan nilai yang terdapat pada Gambar 5. Gambar 8 merupakan tabel hasil *benchmark* yang menampilkan hasil pengujian terhadap sistem *load balancing* metode *weighted round robin*. Tabel ini berisikan jumlah *request* yang diproses *load balancer*, jumlah *request* yang dikirim pada web server 1, jumlah *request* yang dikirim pada web server 2, nilai *response time* web server 1, dan nilai *response time* web server 2. Gambar 9 merupakan grafik batang jumlah *request*. Nilai yang terdapat pada grafik ini sama dengan nilai yang terdapat pada Gambar 8. Gambar 10 merupakan grafik batang nilai *response time*. Nilai yang terdapat pada grafik ini sama dengan nilai yang terdapat pada Gambar 8. Seluruh hasil pengujian ditampilkan dalam bentuk tabel untuk memudahkan melihat hasil pengujian yang telah dilakukan Tabel hasil pengujian dapat dilihat pada Tabel 3 dan Tabel 4.

Tabel 3 Hasil Pengujian Jumlah *Request*

Nama Sampel	Jumlah *Request* Web Server 1 *Round Robin*	Jumlah *Request* Web Server 2 *Round Robin*	Jumlah *Request* Web Server 1 *Weighted Round Robin*	Jumlah *Request* Web Server 2 *Weighted Round Robin*
Sampel Pengujian 1	500	500	333	667
Sampel Pengujian 2	1000	1000	666	1334
Sampel Pengujian 3	1500	1500	1000	2000
Sampel Pengujian 4	2000	2000	1333	2667
Sampel Pengujian 5	2500	2500	1666	3334
Sampel Pengujian 6	3000	3000	2000	4000

Sampel Pengujian 7	3500	3500	2333	4667
Sampel Pengujian 8	4000	4000	2666	5334
Sampel Pengujian 9	4500	4500	3000	6000
Sampel Pengujian 10	5000	5000	3333	6667

Tabel 4 Hasil Pengujian *Response Time*

Nama Sampel	*Response Time* Web Server 1 *Round Robin* (ms)	*Response Time* Web Server 2 *Round Robin* (ms)	*Response Time* Web Server 1 *Weighted Round Robin* (ms)	*Response Time* Web Server 2 *Weighted Round Robin* (ms)	*Response Time* Total RR (ms)	*Response Time* Total WRR (ms)	Persentase Perbedaan *Response Time* Total
Sampel Pengujian 1	3	2	2	3	5	5	0%
Sampel Pengujian 2	3	3	2	3	6	5	17%
Sampel Pengujian 3	3	3	2	4	6	6	0%
Sampel Pengujian 4	6	2	3	4	8	7	13%
Sampel Pengujian 5	5	3	5	3	8	8	0%
Sampel Pengujian 6	4	3	6	3	7	9	22%
Sampel Pengujian 7	7	2	2	4	9	6	33%
Sampel Pengujian 8	6	3	5	4	9	9	0%
Sampel Pengujian 9	11	2	3	4	13	7	46%
Sampel Pengujian 10	10	2	4	3	12	7	42%

Hasil pengujian pada server *load balancing* dengan metode *round robin* berhasil dengan mengirimkan 10 sampel pengujian pada server *load balancing*. Sampel pengujian tersebut telah berhasil diproses oleh *load balancing* yang kemudian mengirimkan *request* tersebut pada web server untuk diproses. *Load balancing* juga telah berhasil menerapkan metode *round robin* dalam pembagian *request* yang dikirimkan pada web server 1 dan web server 2 dimana jumlah *request* yang diproses oleh web server 1 dan web server 2 memiliki nilai yang sama. Akan tetapi, meskipun jumlah *request* yang diproses oleh web server 1 dan web server 2 sama, nilai *response time* dari web server 1 dan web server 2 tidak sama. Nilai *response time* web server 1 jauh lebih besar dibandingkan dengan nilai *response time* web server 2 dengan pengecualian pada sampel pengujian 2 dimana pada pengujian tersebut nilai *response time* web server 1 sebesar 3 ms dan nilai *response time* web server 2 sebesar 3 ms. Ini membuktikan bahwa ketika pengujian dilakukan, web server 2 memiliki kondisi yang jauh lebih baik dibandingkan dengan web server 1. Hal inilah yang menjadi kelemahan dari metode penjadwalan *round robin* dimana metode ini tidak memperdulian

kondisi, kapasitas, dan beban layanan dari server.

Hasil pengujian pada server *load balancing* dengan metode *weighted round robin* berhasil dengan mengirimkan 10 sampel pengujian pada server *load balancer*. Sampel pengujian tersebut telah berhasil diproses oleh *load balancing* yang kemudian mengirimkan *request* tersebut pada web server untuk diproses. *Load balancing* juga telah berhasil menerapkan metode *weighted round robin* dalam pembagian *request* yang dikirimkan pada web server 1 dan web server 2 dimana jumlah *request* yang diproses oleh web server 1 memiliki beban 1 *request* dan web server 2 memiliki beban 2 *request*. Karena beban *request* yang dimiliki setiap web server berbeda, jumlah *request* yang diproses oleh web server juga berbeda. Web server 1 memiliki jumlah *request* yang lebih kecil dibandingkan web server 2 karena beban *request* pada web server ini hanya sebesar 1 *request*. Beban *request* sebesar 2 pada web server 2 *request* menyebabkan jumlah *request* yang diproses web server 2 sebesar 2 kali lipat bahkan lebih dibandingkan web server 1. Hal ini juga yang menyebabkan nilai *response time* pada web server 1 jauh lebih kecil dibandingkan dengan nilai *response time* web server 2 dengan pengecualian sampel pengujian 5, 6, 8, dan 10 dimana pada sampel pengujian tersebut nilai *response time* web server 1 lebih besar daripada web server 2.

Berdasarkan hasil pengujian yang telah dilakukan pada penelitian ini, jumlah *request* yang semakin meningkat pada sampel pengujian 1 sampai dengan sampel pengujian 10 akan menyebabkan nilai *response time* total juga semakin tinggi pada *load balancing* metode *round robin* dengan pengecualian sampel pengujian 6 dan sampel pengujian 10 dimana sampel pengujian tersebut tidak mengalami peningkatan nilai *response time* total dibandingkan dengan sampel pengujian sebelumnya. Pada *load balancing* metode *weighted round robin*, jumlah *request* yang semakin meningkat juga menyebabkan peningkatan nilai *response time* total dengan pengecualian sampel pengujian 7, 9, dan 10 dimana sampel pengujian tersebut tidak mengalami peningkatan nilai *response time* total dibandingkan dengan sampel pengujian sebelumnya.

Berdasarkan hasil pengujian yang telah dilakukan pada penelitian ini, nilai *response time* total dari *load balancing* dengan metode *round robin* dan metode *weighted round robin* memiliki nilai *response time* total yang sama pada sampel pengujian 1, 3, 5, dan 8. Nilai *response time* total *load balancing* dengan metode *round robin* lebih kecil dibandingkan *load balancing* metode *weighted round robin* pada sampel pengujian 6 dimana nilai *response time* total *load balancing* dengan metode *round robin* sebesar 7 ms dan nilai *response time* total *load balancing* metode *weighted round robin* sebesar 9 ms. Nilai *response time* total *load balancing* metode *round robin* lebih besar dibandingkan *load balancing* dengan metode *weighted round robin* pada sampel pengujian 2, 4, 7, 9, dan 10. Pada sampel pengujian 2 *load balancing* dengan metode *round robin* nilai *response time* total sebesar 6 ms dan nilai *response time* total *load balancing* dengan metode *weighted round robin* sebesar 5 ms. Nilai *response time* total *load balancing* dengan metode *weighted round robin* pada sampel pengujian 2 lebih kecil 17% dibandingkan *load balancing* dengan metode *round robin*. Pada sampel pengujian 4 *load balancing* dengan metode *round robin* nilai *response time* total sebesar 8 ms dan nilai *response time* total *load balancing* dengan metode *weighted round robin* sebesar 7 ms. Nilai *response time* total *load balancing* dengan metode *weighted round robin* pada sampel pengujian 4 lebih kecil 13% dibandingkan *load balancing* dengan metode *round robin*. Pada sampel pengujian 7 nilai *response time* total *load balancing* dengan metode *round robin* sebesar 9 ms dan nilai *response time* total *load balancing* dengan metode *weighted round robin* sebesar 7 ms. Nilai *response time* total *load balancing* dengan metode *weighted round robin* pada sampel pengujian 7 lebih kecil 33% dibandingkan *load balancing* dengan metode *round robin*. Pada sampel pengujian 9 nilai *response time* total *load balancing* dengan metode *round robin* sebesar 13 ms dan nilai *response time* total *load balancing* dengan metode *weighted round robin* sebesar 7 ms. Nilai *response time* total *load balancing* dengan metode *weighted round robin* pada sampel pengujian 9 lebih kecil 46% dibandingkan *load balancing* dengan metode *round robin*. Pada sampel pengujian 10 nilai *response time* total *load balancing* dengan metode *round robin* sebesar 12 ms dan nilai *response time* total *load balancing* dengan metode *weighted round robin* sebesar 7 ms. Nilai

response time total *load balancing* dengan metode *weighted round robin* pada sampel pengujian 10 lebih kecil 42% dibandingkan *load balancing* metode *round robin*.

5. KESIMPULAN

Kesimpulan yang diambil dari keseluruhan proses dan pembahasan penelitian yang telah dijelaskan dari bab sebelumnya adalah sebagai berikut:

1. Penerapan sistem *load balancing* dengan metode *round robin* akan menghasilkan *load balancing* yang mendistribusikan beban *request* secara bergiliran dan merata pada web server. Penerapan sistem *load balancing* dengan metode *weighted round robin* akan menghasilkan *load balancer* yang mendistribusikan beban *request* sesuai dengan beban *request* yang diberikan pada web server.
2. Berdasarkan hasil pengujian 10 sampel pengujian dengan nilai *request* yang terus bertambah pada setiap pengujian, semakin besar jumlah *request* yang dikirimkan kepada *load balancer* maka nilai *response time* juga akan semakin besar.
3. Perbandingan nilai *response time* total *load balancing* metode *weighted round robin* dengan beban 1:2 dan *load balancing* metode *round robin* yang mempunyai spesifikasi server yang sama akan menghasilkan selisih persentase *response time* total antara 13% sampai dengan 46%.

6. Saran

Berdasarkan hasil pengujian yang diperoleh dalam penelitian ini, maka diberikan beberapa saran yaitu:

1. Pada penelitian selanjutnya menggunakan jumlah *cluster server* yang lebih banyak dengan spesifikasi yang lebih bervariatif.
2. Pada penelitian selanjutnya membahas analisa lain serta metode penjadwalan lain pada *load balancer*.

DAFTAR PUSTAKA

[1] I. G. L. P. E. Supramana, Prismana, "Implementasi Load Balancing Pada Web Server Dengan Menggunakan Apache," *J. Manaj. Inform.*, vol. 5, pp. 117–125, 2016.

[2] S. D. Riskiono and D. Pasha, "Analisis Metode Load Balancing Dalam Meningkatkan Kinerja Website E-Learning," *J. Teknoinfo*, vol. 14, no. 1, p. 22, 2020, doi: 10.33365/jti.v14i1.466.

[3] A. Rahmatulloh and F. MSN, "Implementasi Load Balancing Web Server menggunakan Haproxy dan Sinkronisasi File pada Sistem Informasi Akademik Universitas Siliwangi," *J. Nas. Teknol. dan Sist. Inf.*, vol. 3, no. 2, pp. 241–248, 2017, doi: 10.25077/teknosi.v3i2.2017.241-248.

[4] B. J. Kuncoro, I. Ruslianto, and S. Bahri, "Jurnal Coding , Sistem Komputer Untan ISSN : 2338-493X IMPLEMENTASI HEALTH CHECKS MONITORING PADA LOAD BALANCER DI SISI WEB SERVER Jurnal Coding , Sistem Komputer Untan Gambar 1 . Arsitektur Umum Load Balancer .," *J. Coding, Sist. Komput. Untan*, vol. 07, no. 01, pp. 33–42, 2019.

[5] S. Malik, "Dynamic load balancing in a network of workstations," *Pap. Parallel Process. Course, Carlet. ...*, no. 219762, 2000, [Online]. Available: http://pdf.aminer.org/000/255/461/customized_dynamic_load_balancing_for_a_network_of_workstations.pdf.

[6] G. Triono, T. Informasi, S. Tinggi, T. Surabaya, and L. V. Server, "Implementasi Load Balancing Dengan Menggunakan Algoritma Round Robin Pada Kasus," *Semin. Nas.*, pp. 169–176, 2015.

[7] D. Chitra Devi and V. Rhymend Uthariaraj, "Load Balancing in Cloud Computing Environment Using Improved Weighted Round Robin Algorithm for Nonpreemptive Dependent Tasks," *Sci. World J.*, vol. 2016, 2016, doi: 10.1155/2016/3896065.

[8] I. Saidu, S. Subramaniam, A. Jaafar, and Z. A. Zukarnain, "A load-aware weighted round-robin algorithm for IEEE 802.16 networks," *Eurasip J. Wirel. Commun. Netw.*, vol. 2014, no. 1, 2014, doi: 10.1186/1687-1499-2014-226.

[9] 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. 01, pp. 109–114, 2019.

[10] HAProxy, "No Title," 2020. http://www.haproxy.org/.

[11] A. Abdullah, S. Simamora, and H. R. Andrian, "Implementasi dan Analisa Load-Balancing pada suatu Web-Server Lokal," *Progr. Stud. Tek. Komput. Politek. TELKOM*, 2010.

[12] K. G. D. Herlangga, "No Title," 2015. https://www.codepolitan.com/belajar-html-dasar.

[13] PHP, "No Title," 2020. https://www.php.net/manual/en/faq.general.php.