Sistem Informatika, Jaringan Komputer, dan Aplikasi
Vol 11 No 2 (2021) **Jurnal GERBANG STMIK Bani Saleh**
ISSN : 0853 - 6376

ANALISIS TROUGHPUT DAN WAKTU RESPON WEB SERVER MENGGUNAKAN LOAD BALANCE DENGAN ALGORITMA ROUND ROBIN

[1]**Domo Pranowo Kuswandono** [2]**Asep Saepuloh**
[1]Program Studi/Jurusan D3 Teknik Komputer, STMIK Bani Saleh
[2]Mahasiswa Program Studi/Jurusan S1 Teknik Informatika, STMIK Bani Saleh
[1]domopranowo@stmik-banisaleh.ac.id, [2]warkirasep.puloh@gmail.com

Abstrak

Load balance merupakan salah satu teknik untuk meminimalisir terjadinya over pada server. Trougput dan *Respond Time* Server menjadi kebutuhan penting untuk saat ini dan diminimalisir untuk terjadinya down, mengingat aplikasi-aplikasi yang dibuat sekarang sudah tersentralisasi pada server. Dengan adanya teknik *load balance* ini maka tiap permintaan klien dibagi rata ke server-server yang tersedia. Adapun Algoritma *Round Robin* adalah salah satu algoritma yang dipakai ketika pembagian permintaan dari klien lalu diteruskan oleh server manajemen secara tersusun. Dengan adanya teknik ini maka peran pertukaran informasi database juga harus sudah kita dukung sehingga perfoamce server bisa optimal.

Kata Kunci: Load balancing, Round Robin, Data Base, Troughput, Respond time.

PENDAHULUAN

Seiring dengan perkembangan teknologi informasi yang semakin cepat, penggunaan web untuk mengakses informasi pun semakin banyak digunakan. Peningkatan permintaan pada situs menyebabkan *web server* sibuk menjawab permintaan-permintaan yang datang dari klien dan sangat memungkinkan *web server* tidak mampu melayani permintaan dari *client* apabila permintaan tersebut sudah melebihi kapasitas . Hal tersebut dapat mengakibatkan *web server* menjadi *overload*, lambat dan akhirnya server menjadi *down*. Hal ini akan merugikan pihak yang mempercayakan situsnya pada suatu *web server*, karena situs-situs tersebut tidak dapat diakses Salah satu untuk mengatasi *overload* perlu mengupgrade *hardware server* ke performa yang lebih tinggi. Namun untuk solusi ini sepertinya hanya akan mengatasi masalah jangka pendek. Salah satu mekanisme untuk mengoptimalkan sumber daya yang ada adalah menggunakan *load balance* yang akan menyeimbangkan beban pada *web server* yang sibuk, sehingga mempercepat waktu respon dari web.

Load Balance adalah terknik untuk mendistribusikan beban trafik pada dua atau lebih jalur koneksi secara seimbang agar trafik dapat berjalan optimal, memaksimalkan *throughput*, memperkecil waktu tanggap dan menghindari overload pada server. *Load Balance* digunakan pada saat sebuah server telah memiliki jumlah user yang telah melebihi maksimal kapasitasnya. *Load Balance* juga mendistribusikan beban kerja secara merata didua atau lebih komputer, link jaringan, *CPU*, hard drive atau sumber daya lainnya untuk mendapatkan pemanfaatan sumber daya yang optimal. Terdapat beberapa metode dalam *Load Balance* seperti *Round Robin*, *Ratio*, *Fastest* Dan *Least Connection.* 2

Round Robin memiliki kelebihan pada pembagian trafik dari klien, algoritma ini bisa membagi rata permintaan klien ke masing-masing server yang tersedia. Pembagiannya secara sekuensial, contohnya jika ada 1 klien melakukan permintaan/*request* page maka permintaan klien tersebut akan diteruskan atau didistribusikan ke server pertama, jika ada permintaan/*request* dari klien 2 maka permintaan tersebut

akan diteruskan oleh server *controller* ke server ke 2, begitupun permintaan selanjutnya. Mudah dalam implementasinya sehingga sangat cocok jika diterapkan pada web server yang berbasis *apache. Ratio* (rasio) sebenarnya merupakan sebuah parameter yang diberikan untuk masing-masing server yang akan dimasukkan kedalam sistem load balancing. Dari parameter Ratio ini, akan dilakukan pembagian beban terhadap server-server yang diberi rasio. Server dengan rasio terbesar diberi beban besar, begitu juga dengan server dengan rasio kecil akan lebih sedikit diberi beban. *Fastest,* Algoritma yang satu ini melakukan pembagian beban dengan mengutamakan server-server yang memiliki respon yang paling cepat. Server di dalam jaringan yang memiliki respon paling cepat merupakan server yang akan mengambil beban pada saat permintaan masuk. *Least Connection,* Algoritma Least connection akan melakukan pembagian beban berdasarkan banyaknya koneksi yang sedang dilayani oleh sebuah server. Server dengan pelayanan koneksi yang paling sedikit akan diberikan beban yang berikutnya akan masuk..

1.1 IDENTIFIKASI MASALAH

Berdasarkan latar belakang yang telah dijelaskan di atas, penulis mengidentifikasikan masalah pada implementasi load balance dengan al gortima *Round Robin* :

1. Dampak Troughout dan R*espon time* kecepatan akses website saat dibuka.
2. Jalur *Backup* yang tidak tersedia ketika server yang utama *down*.
3. Mengkonfigurasi sistem agar dapat memberikan *availability* atau ketersediaan yang baik pada *server* dengan meminimalisir terjadinya *downtime*

1.2 RUMUSAN MASALAH

1. Bagaimana Pengaruh Througput dan Response Time antara yang menggunakan single server dan Load Balance ?
2. Bagaimana cara mengkonfigurasi Server agar bisa berjalan algoritmanya ?
3. Bagaimana cara perhitugan throughtput antara yang single server dengan

1.3 TUJUAN PENELITIAN

Tujuan penelitian Analisis Kecepatan Respon WebServer ini adalah ;

1. Mengetahui dampak respond time server jumlah user yang melakukan akses terhadap webserver
2. Membandingkan performance single web server dengan Multi server dengan Load Balancing
3. Ketersedian Server handal dengan mempersiapkan server backup secara realtime

2. METODE

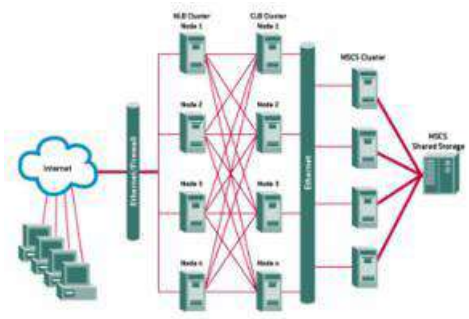
Dalam analisis respond time server ini ada beberapa hal atau kegiatan yang dilakukan,

Menggunakan model Load balancing dengan algoritma Round Robin untuk membagi beban kerja webser.

Dilanjutkan dengan pengaturan konfigurasi load balancing dan web server.

Pengujian terhadap respond time server dengan memberikan percobaan akses dengan jumlah user yang makin banyak. Yang selanjutnya akan di bandingkan dampak jika menggunakan single server.

2.1 LOAD BALANCING



Gambar 2. 1 Cara Kerja *Load Balance*

Load balancing adalah teknik untuk mendistribusikan beban trafik pada dua atau lebih jalur koneksi secara seimbang, agar trafik dapat berjalan optimal, memaksimalkan throughput, memperkecil waktu tanggap dan menghindari overload pada salah satu jalur koneksi. Load balancing digunakan pada saat sebuah server telah memiliki jumlah user yang telah melebihi maksimal kapasitasnya. Load balancing juga mendistribusikan beban kerja secara merata di dua atau lebih komputer, link jaringan, CPU, hard drive, atau sumber daya lainnya,

untuk mendapatkan pemanfaatan sumber daya yang optimal.

Load balancing mengimplementasikan beberapa metode penjadwalan yang menentukan ke arah server mana request dari klien akan diteruskan. Berikut ini merupakan keuntungan yang diperoleh dari teknik *load balancing* :

1. *Flexibility* : Server tidak menjadi inti sistem dan resource utama, tetapi menjadi bagian dari banyak server yang membentuk cluster. Hal ini berarti performa perunit dari cluster tidak terlalu diperhitungkan, tetapi performa cluster secara keseluruhan. Sedangkan untuk meningkatkan performa dan cluster, server atau unit baru dapat ditambahkan tanpa mengganti unit.
2. *Scalability* : Sistem tidak memerlukan desain ulang seluruh arsitektur sistem untuk mengadaptasikan sistem tersebut ketika menjadi perubahan pada komponen sistem.
3. *Security* : Semua traffict yang melewati load balancer, aturan keamanan dapat diimplementasikan dengan mudah. Dengan private network digunakan untuk real server, alamat IP tidak akan diakses secara langsung dari luar sistem cluster.
4. *High-availability :* Load balancer dapat mengetahui kondisi real server dalam sistem secara otomatis, jika terdapat real server yang mati maka akan dihapus dari real server, dan jika real server tersebut aktif kembali maka akan dimasukkan ke dalam daftar real server. (Rabu, Purwadi, & Raharjo, 2012).

2.1.1 Cara Kerja *Load Balance*

Load Balancer (perangkat load balancing) menggunakan beberapa peralatan yang sama untuk menjalankan tugas yang sama. Hal ini memungkinkan pekerjaan dilakukan dengan lebih cepat dibandingkan apabila dikerjakan oleh hanya 1 peralatan saja dan dapat meringankan beban kerja peralatan, serta mempercepat waktu respon. *Load Balancer* bertindak sebagai penengah diatara layanan utama dan pengguna, dimana layanan utama merupakan sekumpulan server/mesin yang siap melayani banyak pengguna.

Disaat *Load Balancer* menerima permintaan layanan dari user, maka permintaan tersebut akan diteruskan ke server utama. Biasanya *Load Balancer* dengan pintar dapat menentukan server mana yang memiliki load yang lebih rendah dan respon yang lebih cepat. Bahkan bisa menghentikan akses ke server yang sedang mengalami masalah dan hanya meneruskannya ke server yang dapat memberikan layanan. Hal ini salah satu kelebihan yang umumnya dimiliki *load balancer*, sehingga layanan seolah-olah tidak ada gangguan di mata pengguna.

2.1.2 Algoritma *Load Balancing*

 Algoritma *Round Robin* merupakan algoritma yang paling sederhana dan banyak digunakan oleh perangkat *load balancing*. Algoritma ini membagi beban secara bergiliran dan berurutan dari satu server ke server lain sehingga membentuk putaran.

- *Ratio (rasio)* sebenarnya merupakan sebuah parameter yang diberikan untuk masing-masing server yang akan dimasukkan kedalam sistem load balancing. Dari parameter Ratio ini, akan dilakukan pembagian beban terhadap server-server yang diberi rasio. Server dengan rasio terbesar diberi beban besar, begitu juga dengan server dengan rasio kecil akan lebih sedikit diberi beban.
- *Fastest*, Algoritma yang satu ini melakukan pembagian beban dengan mengutamakan server-server yang memiliki respon yang paling cepat. Server di dalam jaringan yang memiliki respon paling cepat merupakan server yang akan mengambil beban pada saat permintaan masuk.
- *Least Connection. Algoritma Least connection* akan melakukan pembagian beban berdasarkan banyaknya koneksi yang sedang dilayani oleh sebuah server. Server dengan pelayanan koneksi yang paling sedikit akan diberikan beban yang berikutnya akan masuk.
- *Weighted Round-Robin Allocation*, *algoritma* ini adalah pengembangan dari algoritma round robin, sedikit perbedaannya adalah algoritma ini bisa membagi beban lebih tinggi ke server atau *cluster* yang memiliki *resource* yang lebih besar

2.1.3 Round Robin

Algoritma *Round robin* 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.

Algoritma ini memproses antrian yang digunakan oleh perangkat *load balancing*. Algoritma ini membagi beban dengan cara bergiliran dan berurutan dari satu virtual server ke virtual server lainnya sehingga membentuk putaran. 18

Algoritma ini memproses antrian dan menggilirnya secara bergantian. Proses akan mendapatkan jatah sebesar *time quantum*. Time quantum merupakan batas periode waktu yang diperbolehkan dalam proses yang sedang berjalan dalam sebuah sistem atau jatah waktu yang digunakan dalam pemprosesan data antrian.

Tentu saja proses ini sangatlah adil karena tak ada proses yang diprioritaskan, semua proses mendapatkan jatah waktu yang sama yaitu (1/n) nilai n merupakan proses antrian, dan tak menunggu lama dari (n-1)q dengan q adalah lama 1 quantum. Adapun ketentuan algoritma Round robin adalah sebagai berikut :

1. Jika proses sebelumnya belum selesai maka proses menjadi runnable atau pemrosesan dialihkan ke proses lain atau berikutnya.
2. Jika waktu quantum belum habis dan proses masih berjalan (selesainya operasi I/O), maka proses akan di blocked dan pemrosesan akan dilaihkan ke proses berikutnya.
3. Jika waktu quantum belum habis dan proses telah selesai maka proses diakhiri dan melanjutkan ke proses lainnya.

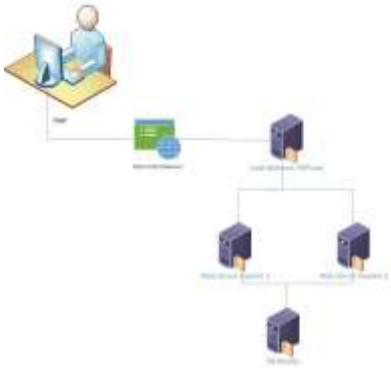
Permasalahan yang terjadi pada algortima *Round robin* adalah menentukan besarnya time quantum. Jika time quantum yang ditentukan terlalu kecil, maka sebagian proses tidak terselesaikan dalam 1 *quantum*. Hal tersebut menyebabkan terjadinya banyak switch (peralihan proses yang terjadi), padahal CPU memerlukan waktu untuk beralih dari satu proses ke proses lainnya *(context switches time)*. Sebaliknya, jika *time quantum* terlalu besar maka, algoritma Round robin akan berjalan seperti algortima *First Come First Served*. *Time quantum* yang ideal adalah 80% dari total proses memiliki CPU *burst time* yang lebih kecil dari 1 *time quantum*. (Mahmood & Rashid, 2011).

2.2 Kebutuhan Perangkat

Spesifikasi minimum perangkat keras yang akan digunakan sebagai server. CPU : Intel Core i5 processor, VGA Card dengan minimum resolusi 1024x768, HDD : 500GB, RAM 8 GB

Kebutuhan perangkat lunak yang dibutuhkan untuk membangun program pada penelitian ini: Server Linux Centos, Virtual Box, HA Proxy untuk Load Balancing, Apache sever dan Database MySql

2.3 Perancangan Sistem

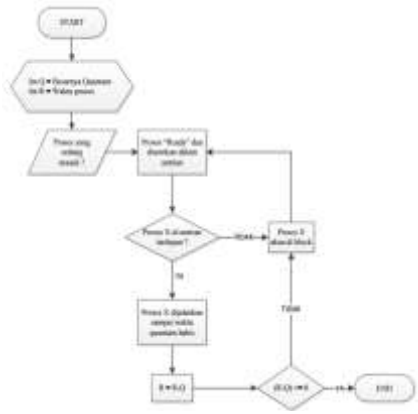


Gambar 3. 1 Kebutuhan Perancangan Jaringan

Pada gambar 3.1 merupakan skenario percobaan dengan menggunakan 2 buah virtual server yang dihubungkan dengan *Director (Load Balancer)*. Masing-masing virtual server menjalankan apache dan sudah di load balance menggunakan server *director*. Pada load balance ini menjalankan perangkat lunak *HAProxy* yang melakukan percobaan penggantian algoritma load balance.

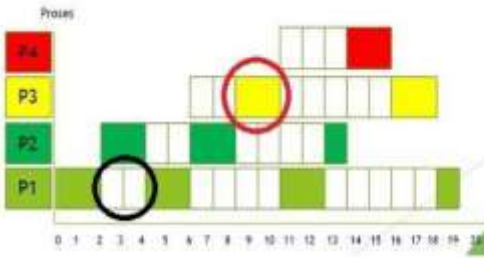
Vol 11 No 2 (2021) **Jurnal GERBANG STMIK Bani Saleh**
ISSN : 0853 - 6376

2.4 Teknik Antrian Round Robin



Dari *flowchart* diatas dapat dijelaskan nilai Q merupakan nilai besarnya quantum dan nilai R merupakan nilai dari waktu pemprosesan. Ketika proses masuk maka proses akan diurutkan dalam antrian (*ready*), jika antrian sudah mencapai antrian terdepan maka proses akan dijalankan sampai waktu quantum habis, jika antrian belum mencapai terdepan maka antrian akan di block dan akan diurutkan kembali ke dalam antrian. Setelah proses antrian berjalan sampai waktu *quantum* habis maka nilai R(waktu proses) = R(waktu proses) – Q(waktu quantum), jika nilai (R-Q) kurang dari atau sama dengan 0 maka proses antrian selesai. Akan tetapi, jika nilai (R-Q) lebih dari atau sama dengan 0 maka proses akan di block dan akan di urutkan kembali ke antrian.Untuk mengetahui bagaimana cara menghitung penjadwalan *Round robin.*

Masing-masing proses memiliki waktu datang (*Arrival Time*) dan waktu proses (*Burst Time*) dengan masing-masing memiliki nilai. Waktu datang merupakan waktu dimana proses mulai mengantri, sedangkan waktu proses (*Burst Time*) merupakan kemungkinan waktu yang digunakan untuk memproses antrian dalam satu proses. Nilai pada Quantum merupakan waktu untuk pemprosesan yang sebenarnya



Dari gambar hasil antrian algoritma *Round robin* tersebut dapat dijelaskan bahwa setiap proses mendapatkan kesempatan waktu quantum sebanyak 2 milisekon, karena P1 >= time quantum maka P1 akan diproses selama waktu quantum, sisa dari P1-time quantum akan diantrikan kembali selanjutnya akan beralih ke P2, P2 akan diproses seperti P1 dan sisanya akan diantrikan kembali dan seterusnya hingga P4 sampai P < time quantum.

Pada algoritma *Round robin* tidak ada proses yang dikerjakan dalam waktu lebih dari time quantum yang disediakan. Jika terdapat n proses dengan 50

time quantum sebesar q, maka masing-masing proses akan mendapatkan waktu 1/n dengan proses sebesar q. Dan proses akan menunggu setidaknya sebanyak (n-1) x q untuk proses selanjutnya. Sebagai contohnya terdapat 4 proses dengan time quantumnya sebesar 20 ms maka setiap proses mendapatkan waktu sebanyak 20 ms setiap 80 ms. Sehingga performa Round robin sendiri tergantung pada ukuran time quantum.

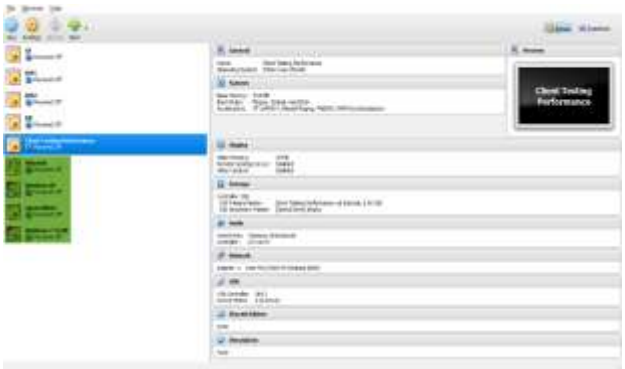
3. HASIL DAN PEMBAHASAN

3.1 Implementasi Software

Perangkat lunak yang dibutuhkan ;

Virtual Box

Virtual Box digunakan untuk melakukan penginstallan beberapa sistem operasi dalam penelitian ini penulis membuat 5 penginstallan sistem operasi maka dalam tahapan ini penulis memberikan contoh ketika pembuatan virtual machine contoh hasil akhirnya akan seperti tampak pada gambar di bawah ini.



Gambar 3. 1 Pemetaan Virtual Machine Virtual Box

Keterangan :

LB / *Load Balancer* = virtual machine yang akan difungsikan sebagai pengatur al goritma yang akan diterapkan.

WS1 (WEB Server 1) dan WS2 (WEB Server 2) = virtual machine yang difungsikan sebagai penerima response request HTTP dari klient.

DB / *Data Base* = virtual machine yang akan difungsikan sebagai database penyimpanan untuk pengetesan performance menggunakan database.

Client Testing Performance = virtual machine ini yang akan difungsikan sebagai klien testing terhadap kinerja al goritma load balance.

HAProxy

HAProxy digunakan untuk mengatur permintaan/*request* yang datang dari klient, di *HAProxy* inilah algoritma round robin berfungsi. Secara otomatis software *HAProxy* akan mengatur mana *request* yang harus dialokasikan ke server 1 dan mana *request* yang harus dialokasikan ke server 2 begitu seterusnya sampai request dari klient terpenuhi. *Versi HAProxy* yang digunakan ketika melakukan penelitian menggunakan *versi HAProxy 1.7.8.*

4.2 Uji *Syncron* Data

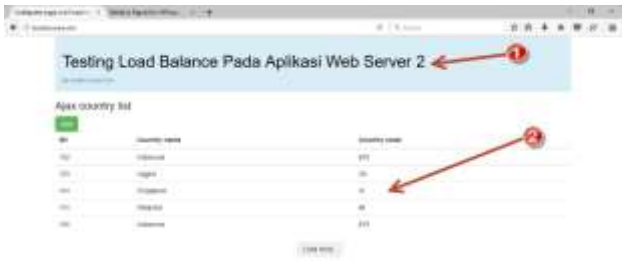
Untuk melakukan pengetesan baik menampilkan data dari database, penginputan data ke *database* penulis mencoba membuat tampilan “webserver 1” dan “webserver 2” aplikasi ini dibuat hanya untuk pembuktian bahwa *Load Balance* bekerja sesuai dengan yang diharapkan.

a. Menampilkan Data

Dalam menampilkan data tujuan yang akan dicapai atau hasil yang akan dicapai yaitu data sama antara web server 1 dengan webserver 2 adapun untuk hasil pengujiannya penulis detailkan antara web server 1 dengan web server 2 dengan mencantumkan identitas pada header program dengan nama identitas “Web Server 1” untuk nama identitas web server 1 dan “Web Server 2” untuk nama identitas web server 2.

Gambar 4.1 Aplikasi Load Balance Di Webserver 1
Keterangan

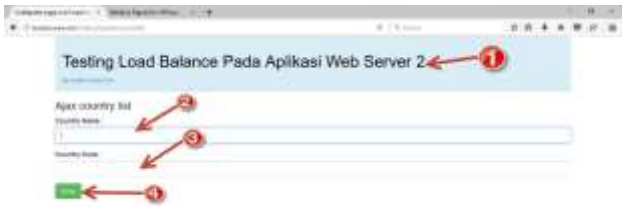
1. Menunjukan bahwa pengaksesan mengarah kepada web server 1
2. Data di webserver 1 ada 5 *record*



Gambar 4.2 Aplikasi Load Balance Di Webserver 2

b. Menambahkan Data

Didalam pengujian hasil menambahkan data penulis menguji menambahkan data pada *web server 2* dengan tujuan ketika melakukan penyimpanan, maka pada *web server* pertama data langsung bertambah sesusai dengan data yang telah ditambahkan pada *web server 2*, pengujian dapat dilihat pada gambar dibawah ini.



Gambar 4. 3 Data Bertambah Pada *Web Server* 1

Keterangan :

1. Nomer satu menunjukan bahwa penambahan data dilakukan di web server 2.

Vol 11 No 2 (2021) **Jurnal GERBANG STMIK Bani Saleh**
ISSN : 0853 - 6376

2. Country Name di*input* data sample contoh "Indonesia".

3. Country Code di*input* dengan kode Negara misalkan "ID".

4. Lakukan penyimpanan dengan mengklick tombol save seperti yang tertera pada nomer 4.

Setelah melakukan penyimpanan data kedalam *Database* maka ketika klien mengakses data akan bertambah data tersebut secara langsung. Dengan pengujian diatas menambahkan data pada *web server 2* ketika *request* diarahkan ke web server 1 data akan bertambah.

Gambar 4. 4 Data Bertambah Pada *Web Server* 1

Keterangan :

1. Menunjukan bahwa identitas *web server* sedang di *webserver* 1

2. Data bertambah yang digambar sebelumnya data hanya 5 *record* setelah di refresh maka data bertambah menjadi 6 *record*

Sehingga ditahap pengujian menampilkan data dari *database* dan menambahkan data ke *database* berhasil. Beban server terbagi secara langsung klien tanpa harus mengetahui *web server* manakah berada. Sebagai pengujian pembuktian *syncron data* berikut tabel rincian nya setelah dilakukan pengujian.

Server	*Header*	Jml Sebelum *Input*	Jml Sesudah Input
WB 1	*Web Server 1*	5	6
WB 2	*Web Server 2*	5	6

Keterangan :

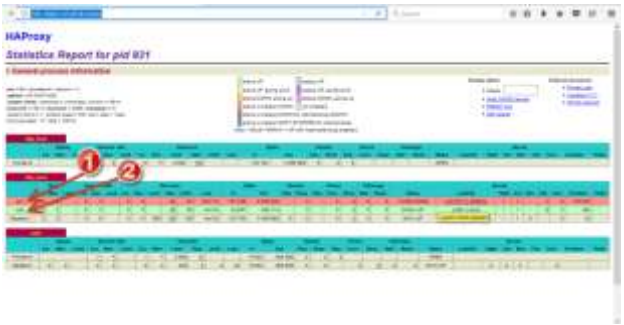
- WB 1 (Web Server 1)
- WB 2 (Web Server 2)
- Header Web Server 1 penamaan identitas web server 1
- Header Web Server penamaan identitas web server 2
- Jml Sebelum Input adalah jumlah data *record* sebelum di *input*

C. Pengujian Mematikan Web Server

Gambar 4. 5 Status Pada Monitoring *Software HAProxy*

Keterangan :

- WS 1 (Web Server 1) Status Hidup dengan label warna hijau
- WS 2 (Web Server 2) Status Hidup dengan label warna hijau



Gambar 4. 6 Status *Web Server 1 Down*

Keterangan :

- WS 1 (Web Server 1) Status *Down* Di *HAProxy* Software dengan label berwarna Merah.

- WS 2 (Web Server 2) Status *Up* Di *HAProxy* Software dengan label berwarna Hijau.

Tabel 4.1 Request Sebelum Dan Sesudah Down Server

Request	*Request Before Down*	*Request After Down*
R1	Web Server 1	*Web Server 2*
R2	Web Server 2	*Web Server 2*

Keterangan :

- R1 (*Request* / Permintaan) 1 Sebelum *Down* di arahkan Ke *Web Server 1*
- R2 (Request / Permintaan) 2 Sebelum *Down* diarahkan ke *Web Server 2*

- R1 (*Request* / Permintaan) 1 Setelah *Down* di arahkan Ke *Web Server 2*

- R2 (*Request* / Permintaan) 1 Setelah *Down* di arahkan Ke *Web Server 2*

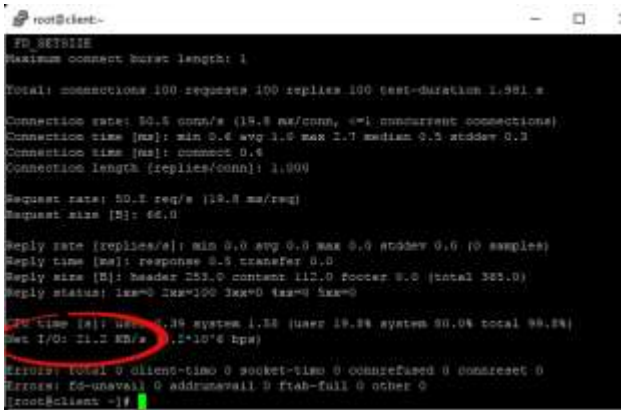
D . Pengujian Single Server dan *Load Balancing*

Dalam penelitian ini penulis hanya melakukan perbandingan dari sisi reponse time, dan throughput yang didapatkan antara *single server* dengan *Load Balance :*

-Throughput

Throughput berdasarkan manual *httperf* dapat diambil dari *NET I/O* yang merupakan rata-rata *throughput* jaringan yang mempunyai satuan *KB (kilobytes)* per detik dan Mb *(megabits)* per detik. Dari sisi throughput akan disajikan beberapa gambar perbandingan antara yang *single server* dengan yang load balance dan akan disajikan table perbandingan untuk memperdetail penelitian.

“httperf --hog --server 192.168.0.141 --num-conn 100 --ra 50 --timeout 5”



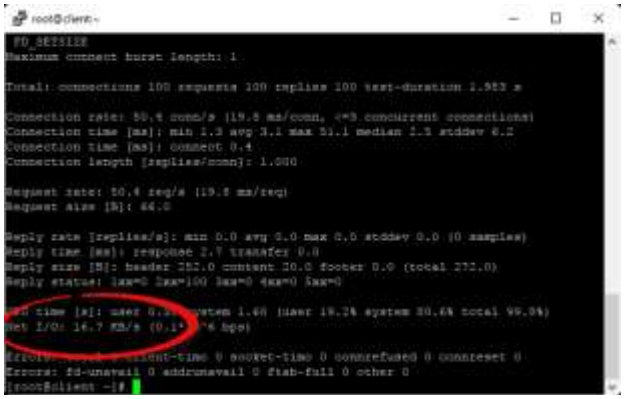
Gambar 4. 7 Throughput Single Server

Keterangan :

- Kecepatan Throughput Net I/O 21.2 KB/s

- -- num-conn 100 artinya membuat 100 koneksi dengan 50 koneksi /detik

- --timeout 5 artinya setelah 5 detik maka pengujian akan diakhiri



Gambar 4.8 Throughput Dengan *Load Balance*

Keterangan :

- Kecepatan Throughput Net I/O 16.7 KB/s

- -- num-conn 100 artinya membuat 100 koneksi dengan 50 koneksi /detik

- --timeout 5 artinya setelah 5 detik maka pengujian akan diakhiri

Dengan pengujian yang dilakukan diatas bahwa sangat terlihat perbandingan throughput yang didapatkan antara yang menggunakan loadbalance dan yang tidak menggunakan *loadbalance* atau *single server* untuk memperdetail berikut dilakukan beberapa percobaan yang akan dicantumkan ditable pengujian throughput

Tabel 4.2 Hasil Uji Troughput

Request/second	Throughput (Kb/Second)	
	Single Server	*Load Balance*
50	*21.4*	*16.8*
100	*21.2*	*16.7*
150	*21.2*	*16.6*
200	*21.1*	*16.6*
250	*21.1*	*16.6*
300	*21.1*	*16.6*
350	*21.1*	*16.6*
400	*21.1*	*16.6*
450	*21.1*	*16.6*
500	*21.1*	*16.6*
550	*21.1*	*16.6*
600	*21.1*	*16.6*
650	*21.1*	*16.5*
700	*21.1*	*16.5*
750	*21.1*	*16.5*
800	*21.1*	*16.5*
850	*21.1*	*16.5*
900	*21.1*	*16.5*
950	*21.1*	*16.5*
1000	*21.1*	*16.5*
10000	*21.0*	*16.5*
Rata-rata	*21.1*	*16.5*

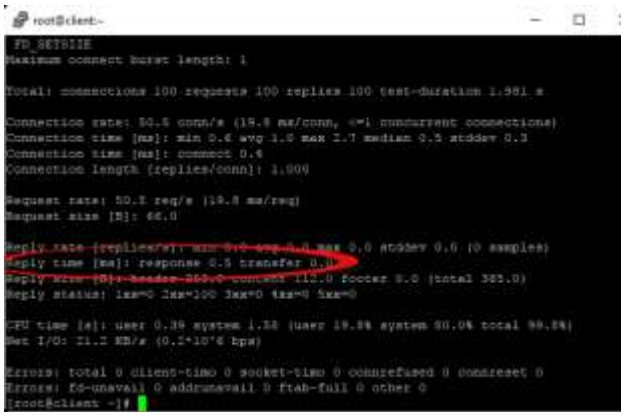
Vol 11 No 2 (2021) **Jurnal GERBANG STMIK Bani Saleh**
ISSN : 0853 - 6376

Pada tabel 4.2 menunjukkan perbandingan dari *throuhgput* yang dihasilkan pada penelitian ini. Implementasi *single server* dengan menggunakan algoritma *Round robin* memiliki rata-rata *throughput* yang lebih baik dibandingkan dengan *single server* pada seluruh implementasi *load balancing* dan tanpa *loadbalancing*. Nilai *throughput* pada hasil penelitian diatas 60 menunjukkan berbanding lurus dengan jumlah *server* yang digunakan, hal ini membuktikan bahwa penambahan jumlah *server* dapat meningkatkan nilai throughput implementasi *load balancing server*.

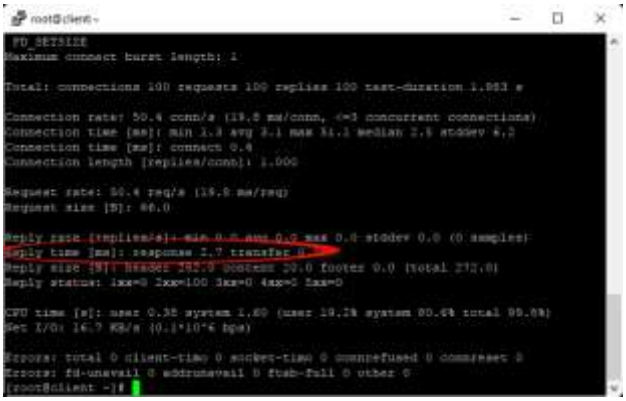
Hasil Perbandingan = Jumlah rata – rata *single server* - Jumlah rata-rata **LB** maka selisih antara kedua perbandingan tersebut H = 21.1 – 16.5 = 4.6 Kb/s maka selesihnya didapat 4.6 Kb/s.

-Response Time

Parameter *response time* pada penelitian ini menggambarkan kecepatan *web server* dalam menanggapi *request* dari *client*. Parameter ini dihitung dengan satuan *milisecond*. Semakin kecil nilai dari parameter ini, maka semakin cepat sebuah *web server* dalam menanggapi *request* dari *client*.



Gambar 4.9 Respond time Single Server



Gambar 4.10 Response Time Dengan *Load Balance*

Tabel 4.3 Hasil Uji Parameter Response Time

Request/second	Response Time (milisecond)	
	Single Server	*Load Balance*
50	*0.5*	*2.7*
100	*0.6*	*1.6*
150	*0.6*	*1.9*
200	*0.7*	*1.5*
250	*0.7*	*2.2*
300	*0.6*	*2.0*
350	*0.5*	*1.6*
400	*0.6*	*1.6*
450	*0.6*	*1.6*
500	*0.6*	*1.6*
550	*0.6*	*1.6*
600	*0.6*	*2.2*
650	*0.6*	*1.6*
700	*0.7*	*1.6*
750	*0.6*	*2.0*
800	*1.6*	*1.6*
850	*0.6*	*1.6*
900	*0.6*	*1.8*
950	*0.8*	*2.1*
1000	*0.7*	*2.1*
10000	*0.7*	*1.7*
Rata-rata	*0.68*	*1.82*

Tabel 4.3 menunjukkan perbandingan dari *response time* yang dihasilkan. Implementasi *load balancing server* dengan membandingkan algoritma sebelum dan sesudah implementasi dengan algoritma *Round robin.* Pada single server sebelum implementasi yaitu dengan memiliki *response time* yang lebih baik dibandingkan dengan algoritma *Round robin* pada implementasi dengan 2. Tetapi peningkatan yang terjadi tidak terlalu signifikan.

PENUTUP

Simpulan

Berdasarkan penelitian yang telah dilakukan maka dapat ditarik kesimpulan sebagai berikut :

1. Dari hasil impementasi uji Throughput algoritma round robin lebih cepat dengan perbedaan 4.6 Kb/s.
2. Response time algoritma round robin dibanding single server, lebih lambat 1.14 milidetik
3. Dalam penelitian ini dibuktikan load balancing round robin mampu meningkatkan kehandalan (avaibility), dengan melakukan simulasi yang telah dilakukan ketika terjadinya *down* pada salah satu web server maka web server yang lain bisa menghandle terhadap permintaan klien sebagai jalur backup

Saran

Berikut adalah saran untuk pengembangan lebih lanjut terhadap penelitian ini sebagai berikut:

- Untuk meningkatkan respond time server bisa dilakukan pengujian dengan melakukan hardware yang lebih tinggi spesifikasinya
- Studi dampak Load balancing terhadap bandwidth yang tersedia sangat bermanfaat utk menentukan berapa besar utilitas dari jalur yang telah ada

DAFTAR PUSTAKA

[1] Bullock, T., 2007. *Web Workload Generator Quickstart Guide*. Canada : Calgary.

[2] Dewobroto, Pujo. 2009, *Load Balance menggunakan metode PCC*.[Online]. Terdapat di : http;//www.mikrotik.co.id/artikel_lihat.php?id=34 [Diakses pada 13 Mei 2021.

[3] Han, Z. & Pan, Q., 2012. *LVS Cluster Technology in the Research and Application of State-owned Asset Management Platform. The 2nd International Conference on Computer Application and System Modeling (2012). ISSN : 1951-6851*

[4] Jogiyanto, 2005. *Sistem Informasi Berbasis Komputer*. Yogyakarta : Andi.

[5] Kurniawan, Y., Sabriasyah & Sakti, E., 2013. *Analisis Kinerja Algoritma Load Balancer dan Implementasi pada Layanan Web*. Universitas Brawijaya, Volume III.

[6] Mahmood, A. & Rashid, I., 2011. *Comparison of Load Balancing Algoritms for Cluster Web Server*. Malaysia, IEEE. ISBN : 978-1-4577-0988-3.

[7] Mulay, S. & Jain, S., 2013. *Enhanced Equally Distributed Load Balancing Algoritm For Cloud Computing. International Journal of Research in Engineering and Technology*