E-ISSN : 3025-1311
https://ejournal.warunayama.org/kohesi

Kohesi: Jurnal Multidisiplin Saintek
Volume 2, No. 9 2024, pp 25-42

IMPLEMENTASI RASPBERRY PI SEBAGAI CLUSTER SERVER MENGGUNAKAN METODE LOAD BALANCE

Achmad Nuril Fauzi
Universitas Muhammadiyah Malang
ahmadnuril96@gmail.com

Abstrak

Tingginya era perkembangan dalam dunia teknologi maupun digital pada masa saat ini yang tumbuh sangat cepat dan sangat pesat, ketersediaan website menjadi semakin krusial sebagai penunjang sarana dan prasarana di berbagai industri. Situs web ini dihosting di server web yang menyediakan layanan berupa data melalui HTTP atau HTTPS. Web server yang handal tentunya mampu melayani request dari pengguna dalam jumlah yang cukup besar dalam satu satuan waktu. Pengujian yang pertama adalah 100 sample dan didapati hasil untuk mencapai 100 samples waktu maksimal yang diperlukan adalah 5248ms dengan rara - rata waktu yang dibutuhkan adalah 2617ms. Pengujian selanjutnya dengan 300 sample didapatkan hasil bahwa waktu yang dibutuhkan pada 300 samples adalah 14083ms dengan rata rata waktu yang dibutuhkan untuk mengakses adalah 8377ms. Pengujian dengan 500 sample didapatkan bahwa untuk 500 sample maksimal adalah 19747ms dan untuk rata – rata waktu yang diperlukan adalah 12000ms. Selanjutnya menggunakan 1000 sample didapatkan waku maksimal yang dibutuhkan untuk menyelesaikan 1000 request client adalah 33749ms dengan rata-rata waktu adalah 19729ms. Setelah pengujian menggunakan metode load balance selesai maka pengujian selanjutnya adalah menguji *raspberry* yang digunakan sebagai server secara individu atau standalone dengan sample data yang digunakan sama seperti pada sample loadbalance. Pertama yang diujikan adalah 100 sample didapatkan hasil waktu maksimal yang dibutuhkan adalah 6869ms dimana dengan 100 sample tersebut rata-rata waktu yang dibutuhkan 5585ms. Selanjutnya menggunakan 300 sample dapat diketahui bahwa pada tabel 4.12 waktu yang dibutuhkan ketika 300 sample di coba adalah 21096ms dengan rata-rata waktu yang dibutuhkan 12948ms. Selanjutnya terpadat pengujian dengan sample 500 ditunjukkan dengan tabel 4.13. dari gambar tersebut dapat diketahui bahwa waktu untuk 500 sample maksimal 33377ms dengan waktu rata – rata yang dibutuhkan 20540ms. Terakhir menggunakan 1000 sampel fdata didapatkan waktu maksimal yang digunakan untuk menyelesaikan adalah 64413ms dengan rata – rata waktu 36281ms.

Kata kunci : web server, raspberry pi, cluster server,loadbalance

ABSTRACT

In the current era of high development in the world of technology and digital which is growing very quickly and very rapidly, the availability of websites is becoming increasingly

crucial as a support for facilities and infrastructure in various industries. This website is hosted on a web server that provides services in the form of data via HTTP or HTTPS.. Along with the development of user needs and increased demand for a website, the work of the web server becomes heavier. A reliable web server is of course capable of serving requests from a fairly large number of users at one time. The first test was 100 samples and it was found that to reach 100 samples the maximum time needed was 5248ms with the average time needed was 2617ms. Subsequent testing with 300 samples showed that the time required for 300 samples was 14083ms with an average access time of 8377ms. Testing with 500 samples found that for 500 samples the maximum is 19747ms and for the average time required is 12000ms. Furthermore, using 1000 samples, the maximum time needed to complete 1000 client requests is 33749ms with an average time of 19729ms. After testing using the load balance method is complete, the next test is to test the raspberries used as individual or standalone servers with the sample data used the same as the load balance sample. The first thing tested was 100 samples, the maximum time needed was 6869ms where with 100 samples the average time needed was 5585ms. Furthermore, using 300 samples, it can be seen that in table 4.12 the time needed when 300 samples are tested is 21096ms with an average time required of 12948ms. Furthermore, the densest test with a sample of 500 is shown in table 4.13. From the figure it can be seen that the maximum time for 500 samples is 33377ms with an average time required of 20540ms. Lastly, using 1000 fdata samples, it was found that the maximum time used to complete was 64413ms with an average time of 36281ms.

Keywords : web server, raspberry, cluster server, load balance

Pendahuluan

Tingginya era perkembangan dalam dunia teknologi maupun digital pada masa saat ini yang tumbuh sangat cepat dan sangat pesat, ketersediaan website menjadi semakin krusial sebagai penunjang sarana dan prasarana di berbagai industri. Situs web ini dihosting di server web yang menyediakan layanan berupa data melalui HTTP atau HTTPS. Tugas server web menjadi lebih sulit karena permintaan pengguna dan permintaan untuk situs web berkembang. Web server yang handal tentu saja dapat memenuhi permintaan dari sejumlah besar pengguna pada saat yang bersamaan [1]. Ketika suatu sistem menerima jutaan permintaan per hari, dapat menyebabkan kinerja sistem menjadi sangat buruk karena kelebihan beban, yang dapat menyebabkan beberapa kesulitan, salah satunya adalah keluhan pengguna [2]. Akibatnya, sistem yang menggabungkan banyak perangkat server web yang beroperasi bersama diperlukan untuk menangani peningkatan jumlah permintaan yang dibuat oleh pengguna.

Teknik yang digunakan dalam mendistribusikan beban lalu lintas secara merata melalui dua atau lebih jalur koneksi adalah penyeimbangan muatan sehingga lalu lintas dapat mengalir seefisien mungkin untuk memaksimalkan throughput, mengurangi waktu respon, dan menghindari kelebihan beban di salah satu koneksi. Dalam penggunaanya load balancing digunakan ketika kapasitas server dilampaui oleh jumlah pengguna. Selanjutnya, sistem load balancing akan membagi beban pada

semua node yang ada secara merata, koneksi jaringan, control prosesing unit atau CPU, penyimpanan , atau perangkat pendukung lainnya untuk memaksimalkan penggunaan server[3]. Oleh karena itu, load balancing membutuhkan sejumlah perangkat yang telah disiapkan untuk bekerja sama. Perangkat mikrokontroler modern mampu menangani kebutuhan website, termasuk sebagai web server, berkat peningkatan performa yang dibawa oleh kemajuan zaman.

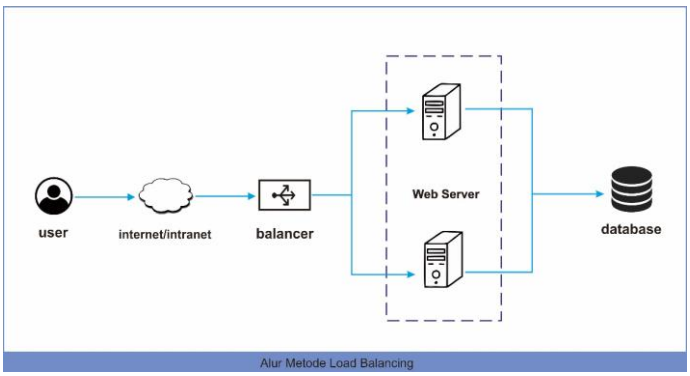
Raspberry Pi adalah ciptaan Raspberry Pi Foundation, sebuah organisasi nirlaba Inggris yang didirikan dengan tujuan untuk menginspirasi kaum muda untuk mengejar karir dalam pengembangan perangkat keras dan perangkat lunak. Raspberry Pi tersedia untuk umum di bawah lisensi Open-Source Hardware, memungkinkan studi, modifikasi, distribusi, perakitan, dan penjualan tanpa batas sesuai dengan desain aslinya. Raspberry Pi langsung mendapatkan popularitas dan telah digunakan untuk berbagai hal, termasuk menjalankan pusat media, komputer jaringan, dan perangkat lunak server web, berkat distribusinya di bawah lisensi Perangkat Keras Sumber Terbuka [4]. Diharapkan load balancing dengan menggunakan Raspberry Pi akan memiliki respon peningkatan kualitas yang lebih tinggi sebagai hasil dari berbagai inovasi Raspberry Pi yang telah dipasok oleh perusahaan, khususnya dalam peningkatan kinerja.

Salah satu penelitian yang pernah dilakukan untuk mendapatkan hasil kinerja yang baik pada web server adalah penelitian D. T. Nugrahadi et al. Berdasarkan temuan penelitian ini yang berjudul Pengaruh Load Balancing dan Web Server Tuning pada Response Time Raspberry Pi, diperoleh response time pada arsitektur tanpa menggunakan load balancing sebesar 2331.4, 2064, dan 1869.2ms dan pada 600 permintaan yang dilayani oleh server web dihasilkan 2202ms . Selanjutnya bisa disimpulkan bahwa web server yang telah menggunakan penyeimbangan muatan memiliki waktu respons yang lebih cepat daripada server web yang tidak menggunakan penyeimbang muatan. Kemudian pengujian tuning dengan 1000permintaan menghasilkan waktu respons 3103,4ms. [5].

Penulis akan menggunakan load balancing untuk membangun server clustering berdasarkan backdrop yang telah diberikan. Sebagai penyeimbang sistem kerja, load balancing memastikan jika satu server down, yang lain segera mengambil alih. Untuk membuat web server yang lebih handal dan mampu menangani permintaan pengguna, penulis menggunakan raspberry pi sebagai web server menggunakan algoritma round-robin.

Metode

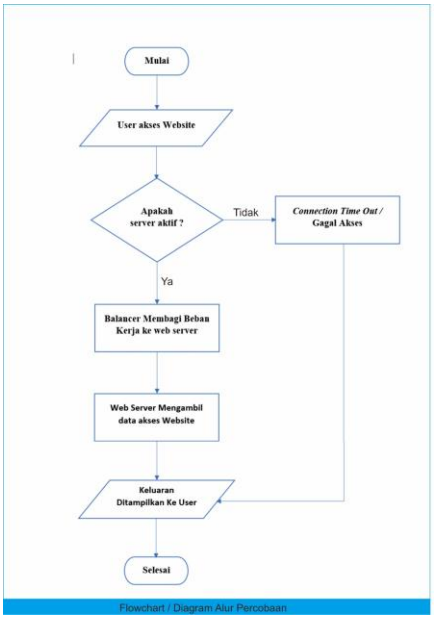
Pada penelitian ini metode yang digunakan adalah *load balance HAproxy* dengan tujuan menggabungkan kinerja dari raspberry pi agar kinerja dari raspberry pi lebih membaik. Alur permodelan pada penelitian yang dilakukan di ilustrasikan pada gambar dibawah ini :



Gambar 3. 1 permodelan penelitian

Dapat dijelaskan pada gambar 3.1, Pada permodelan penelitian diatas dapat diuraikan bahwa pada penelitian ini dimulai dari user melakukan akses *HTTP* atau *HTTPS* sebuah *web server* dengan memalui internet maupun intranet kemudian permintaan tersebut diterima oleh *load balancer*. Setelah data diterima oleh *load balancer* permintaan tersebut akan diteruskan ke *web server* dengan mempertimbangkan beban dari *web server* atau dengan kata lain sebuah *balancer* akan menyeimbangkan kinerja dari beberapa *web server*. setelah user mendapatkan akses ke website yang dituju, data akan disimpan sebuah *database server*.

Pada penelitian ini juga terdapat diagram alir atau flowchart dimana untuk mempermudah dalam pembacaan sistem, untuk flowchart dari percobaan ini dapat ditunjukkan pada gambar berikut :



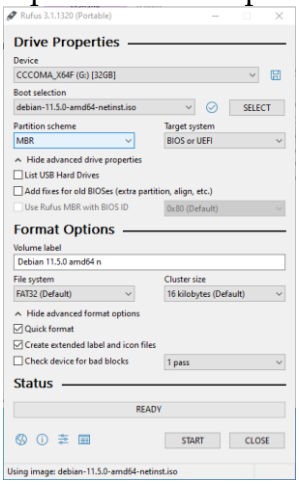
Gambar 3. 2 flowchart program

Dari gambar 3.2 Diatas dapat dijelaskan bahwa awal mulai percobaan adalah ketika user mengakses sebuah website dimana permintaan tersebut akan dilakukan pengecekan apakah web server aktif atau tidak, jika aktif maka dilanjutkan dengan balancer akan membagi kerja webserver raspberry kemudian web server akan mengakses direktori dari website kemudian ditampilkan pada user dan jika web

server keadaan tidak aktif maka akan terjadi connection time out / gagal kemudian kode gagal akan ditampilkan kepada user dan percobaan selesai.

4.1 Konfigurasi operasi sistem

Untuk konfigurasi operasi sistem pada balancer dan backend server raspberry, diperlukan file iso debian versi 11 atau bullseye yang digunakan pada sistem operasi balancer dan file img ubuntu versi 22.04 atau jammy jellyfish. Untuk melakukan penginstalanpun berbeda dimana untuk balancer server iso file di tulisan ke dalam flash disk sedangkan pada server raspberry file img akan dicopykan terlebih dahulu kedalam microSD sebagai sistem penyimpanannya. Untuk penulisan atau bootable pada sistem balancer digunakan aplikasi rufus seperti gambar dibawah ini :



Gambar 4.1 membuat bootable debian 11

Pada gambar 4.1 dapat diketahui pada aplikasi rufus device yang digunakan adalah flashdisk dengan kapasitas 32GB, untuk boot atau iso yang digunakan adalah debian 11.5.0-amd64-netisnt.iso. setelah itu start dan tunggu hingga proses bootable selesai. Setelah selesai flashdisk siap untuk dijadian bootable pada balancer. Dalam penginstallan sistem operasi pada balancer penginstallanya sama seperti penginstallan sistem operasi pada linux, dimana tinggal atur boot priority ke usb lalu install sesuai arahan yang ada pada layar monitor. Berbeda dengan sistem operasi yang di instalkan pada raspberry atau backend server dimana untuk dapat dilajankan pada raspberry file sistem img haruslah dituliskan dulu pada raspberry pi mananger, untuk tampilan raspberry pi mananger seperti berikut :

Gambar 4.2 raspberry pi mananger

Pada gambar 4.2 terdapat tampilan raspberry pi manager dimana pada tampilan tersebut diketahui bahwa file img yang akan di install adalah 20220121_RASPI_3_BULLSEYE.IMG dan untuk penyimpanan yang digunakan adalah microSD yang sebelumnya sudah dimasukkan kedalam cardreader agar terbaca oleh sistem pada laptop. Setelah penulisan ke dalam microSD selesai, lepaskan microSD dari cardreader dan pasang ke dalam raspberry pi, tunggu dan konfigurasi sesuai perintah yang terdapat dilayar.

4.2 Konfigurasi haproxy

Setelah operasi sistem pada *balancer* maupun *backend server* atau *raspberry* telah di lakukan penginstallan maka konfigurasi dilanjukan pada sisi balancer hingga semua terkonfigurasi atau terhubung antara *balancer* dengan *backend server.* Hasil dari tahapan penggabungan raspberry hingga menjadi satu-kesatuan sebagai *backend* ditunjukan oleh tabel 4.1.

Tabel 4.1 Hasil konfigurasi *backend server* dengan raspberry

	Server							
	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle
Raspberry01	L4OK in 1ms	1	Y	-	0	0	0s	-
Raspberry02	L4OK in 1ms	1	Y	-	0	0	0s	-
Backend		2	2	0		0	0s	

Dari tabel 4.1 diatas dapat diketahui bahwa status dari raspberry 01 maupun rasppberry 02 bertasus L4OK in 1ms yang dimana berati status dari backend server dalam keadaan aktif. Pada kolom backend juga dapat diketahi bahwa bayaknya backan pada percobaan ini adalah 2.

Proses selanjutnya setelah dari konfigurasi haproxy adalah pengujian haproxy. Tahap ini merupakan tahapan apakah haproxy telah berjalan dengan benar sesuai dengan apa yang di inginkan. Pada proses pengujian ini *server balancer* akan dilakukan *refresh* halana beberapa kali hingga secara bergantian menampilakn halaman web dari sisi *backand atau raspberry.* Proses memuat data website tersebut dilakukan secara merata dan diatur oleh *server balancer* yang sebelumnya telah dikonfigurasikan dengan menggunakan algoritma *roundrobin* pada kinerjanya.

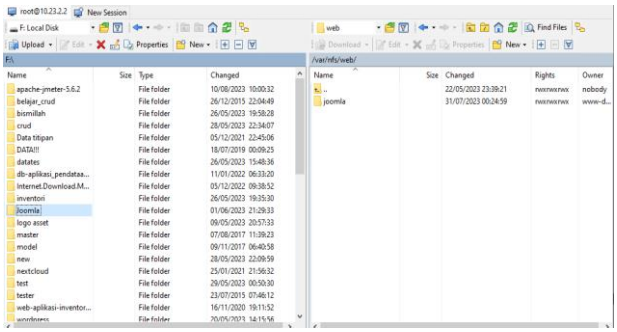
Tabel 4.2 data *Sessions* pada *balancer*

	Sessions						Bytes	
	Cur	Max	Limit	Total	LbTot	Last	In	Out
Raspberry01	0	1	-	39	39	2s	16 306	17 004
Raspberry02	0	1	-	39	39	2s	16 302	17 004
Backend	0	1	26 212	78	78	2s	32 608	34 008

Dari data tabel yang tersaji pada tabel 4.2 dapat diketahui bahwa proses percobaan yang telah dilakukan sebanyak 78 kali dimana pada percobaan tersebut *request* yang dilakukan oleh *user* didistribusikan secara merata melalui kedua *backand raspberry* aktif dengan masing masing *handle request client* sebanyak 39 kali. Dari hasil ini dapat diketahui bahwa algoritma pada sistem *loadbalance* telah sesuai dengan yang di inginkan.

4.3 Data webserver

Seletah hasil uji pada raspberry berhasil dilakukan maka selanjutnya adalah melakukan *upload* data webserver berupa website. Pada proses ini website yang digunakan sebagai sampling adalah joomla, proses upload ini menggunakan protokol ssh dengan menggunakan aplikasi winSCP untuk memudahkan dalam proses *uploading* data. Data dari website tersebut akan dimasukkan kedalam direktori /var/nfs/web/ seperti pada gambar 4.3 dibawah ini.



Gambar 4.3 Proses uload data website joomla.

Pada gamabar 4.3 terdapat 2 tampilan yang berbeda dimana pada sisi kiri adalah penyimpanan yang terdapat pada sistem komputer dan pada sisi kanan adalah isi penyimpanan yang terdapat pada server raspberry, selanjutnya dapa joomla akan dimasukkan dari file yang berapada pada komputer menuju direktori /var/nfs/web yang ada pada sisi raspberry dengan cara klik dan tarik folder jomla menuju folder yang dituju dan tunggu hingga semua file berhasil tersalin.

4.4 Mount file

Setelah file website sudah berhasil diunggah maka selanjutnya adalah proses *mounting* file website yang berada pada *balancer* ke tiap-tiap *node* raspberry sehingga pada saat pengguna melakukan *request* data ke *balancer* maka isi dari tiap raspberry sama. Penggunaan sistem ini juga bertujuan agar lebih efisiensi penggunaan penyimpanan dan juga untuk mengurasi resiko dari file website yang berbeda-beda ketika dimasukkan kedalam data sistem raspberry masing-masing. Konfigurasi hasil *mounting raspberry* terhadap balancer dan juga letak konfigurasi direktori dapat dilihat pada gambar 4.2.

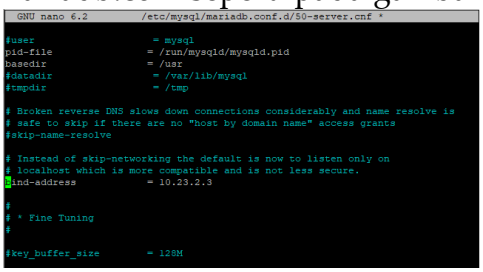
```
root@RaspberrySkripsi-01:~# df -h
Filesystem              Size  Used Avail Use% Mounted on
tmpfs                    92M  3.0M   89M   4% /run
/dev/mmcblk0p2           29G  3.4G   24G  13% /
tmpfs                   459M     0  459M   0% /dev/shm
tmpfs                   5.0M     0  5.0M   0% /run/lock
/dev/mmcblk0p1          253M   67M  186M  27% /boot/firmware
10.23.2.2:/var/nfs/web   55G  1.6G   51G   3% /nfs/web
```

Gambar 4.4 *Mount point* pada *backend raspberry*

Pada gambar 4.4 diketahui bahwa terdapat beberapa mounting saat raspberry server dihidupkan. Untuk mount point yang paling bawah adalah mount point yang telah dibuat, dimana mounting tersebut pada ip 10.23.2.2 pada direktori var/nfs/web dan di letakkan pada /nfs/web.

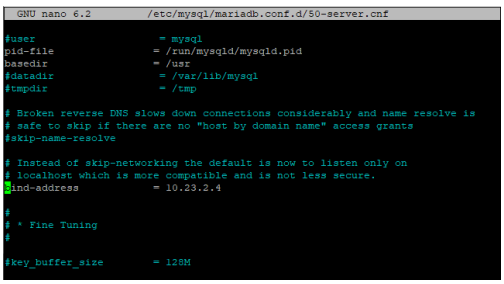
4.5 Replication database

Setelah semua proses dari *haproxy* dan *upload* data selesai, proses selanjutnya adalah membuat *database* untuk hasil data dari konfigurasi joomla. Proses pembuatan database ini dilakukan dengan menggabungkan *backend node raspberry*. Dengan demikian maka database akan terinstal pada semua *raspberry* dengan ketentuan data yang dihasilkan akan sama antara satu *raspberry* dengan *raspberry* lainnya dalam keadaan semua *backend* hidup. Dari skematik yang telah dibuat pada bab 3 dimana ada 2 node yang bertindak sebagai master to master, dimana pada skema ini ketika salah satu server keadaan *downtime* maka data akan otomatis tersimpan pada server yang berstatus *state* atau bekerja dan ketika server *downtime* kembali *state* atau bekerja maka data otomatis tersingkronisasi. Untuk dapat melakukan konfigurasi replication database, hal yang perlu dilakukan adalah melakukan penginsallan aplikasi yang dibutuhkan yaitu database. Database yang digunakan pada percobaan ini adalah mariadb. Setelah database berhasil di install hal yang perlu dilakukan adalah melakukan konfigurasi. Untuk masuk ke konfigurasi dengan membuka file 50-server.cnf pada /etc/mysql/mariadb.conf seperti pada gambar dibawah ini.



Gambar 4.5 konfigurasi mysql pada database S1

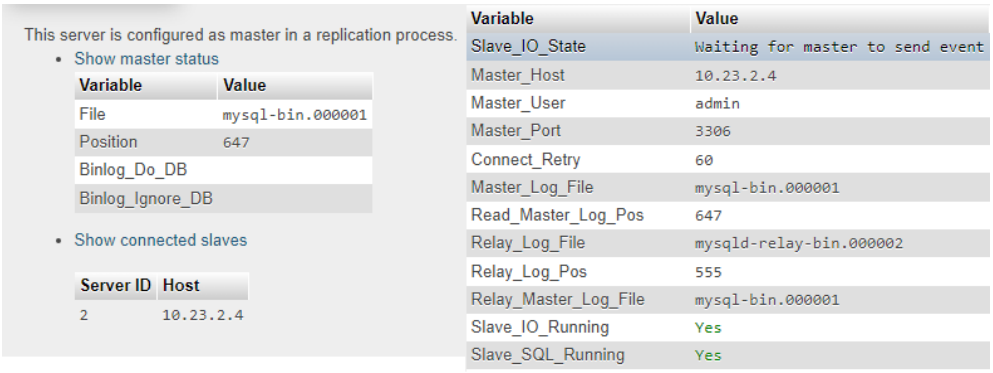
Pada gambar diatas adalah isi dari file 50-server.cnf dimana pada file ini terdapat konfigurasi yang berhubungan dengan database replication. Pada gambar 4.5 diatas dapat diketahui bahwa pada bagian bind-address telah disesuaian dengan ip backend server raspberry yaitu pada 10.23.2.3. secara default ip address pada bind-address adalah 0.0.0.0. setelah konfigurasi pada server S1 selesai, dilanjutkan dengan konfigurasi pada server S2 dimana untuk membuka isi file tersebut juga sama seperti konfigurasi pada server S2. Untuk tampilkan file 50-server.cnf seperti gambar berikut :



Gambar 4.6 konfigurasi mysql pada database S2

Dari gambar 4.6 tersebut dapat diketahui bahwa bind-address nya juga sudah disesuaikan dengan penggunaan ip pada server S2 dimana ipnya telah berubah yang sebelumnya 0.0.0.0 menjadi 10.23.2.4. pada file ini baik pada konfigurasi server S1 dan server S2 juga pada bagian *server-id* dan *log-bin* masih terdapat *comment* atau tanya pagar, maka perlu dilakukan *uncomment* atau menghapus simbol pagarnya simpan

dan restart mysql. Setelah konfigurasi selesai dilakukan pengetesan dilanjutkan dengan web gui yaitu dengan memanfaatkan aplikasi phpmyadmin. Untuk dapat membuka aplikasi phpmyadmin ini dibutuhkan broswer untuk membukanya yaitu dengan mengetikan http://alamatipserver/phpmyadmin. Sebagai contoh pada percobaan ini adalah http://10.23.2.3/phpmyadmin lalu login dengan username dan password yang dibuat lalu masuk pada bagian repliction maka akan muncul tampilan seperti dibawah ini.


This server is configured as master in a replication process.

- Show master status

Variable	Value
File	mysql-bin.000001
Position	647
Binlog_Do_DB	
Binlog_Ignore_DB	

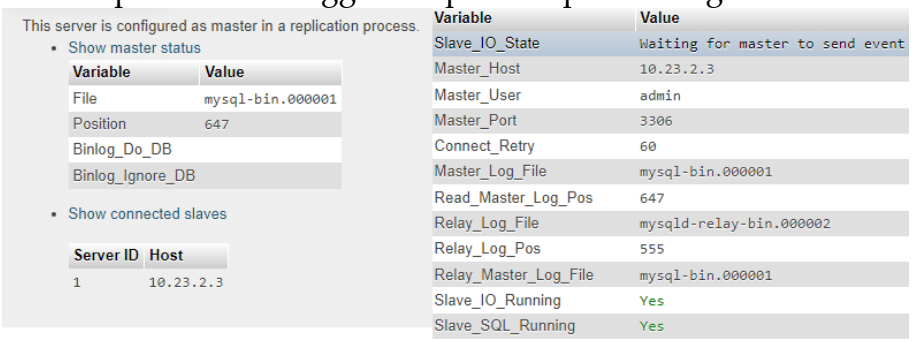
- Show connected slaves

Server ID	Host
2	10.23.2.4

Variable	Value
Slave_IO_State	Waiting for master to send event
Master_Host	10.23.2.4
Master_User	admin
Master_Port	3306
Connect_Retry	60
Master_Log_File	mysql-bin.000001
Read_Master_Log_Pos	647
Relay_Log_File	mysqld-relay-bin.000002
Relay_Log_Pos	555
Relay_Master_Log_File	mysql-bin.000001
Slave_IO_Running	Yes
Slave_SQL_Running	Yes

Gambar 4.7 Status *master-slave* pada *raspberry* 1 atau S1

Pada gambar 4.7 menunjukan bahwa pada *raspberry* 1 atau S1 menjadi master dari IP 10.23.2.4 (S2). Pada gambar tersebut juga dapat diketahui bahwa *raspberry* 1 bertindak sebagai slave dari IP 10.23.2.4 (S2). Selanjutnya buka web gui pada phpmyadmin pada server S2 di alamat http://10.23.2.4/phpmyadmin kemudian login dan buka replication sehingga didapati tampilan sebagai berikut.


This server is configured as master in a replication process.

- Show master status

Variable	Value
File	mysql-bin.000001
Position	647
Binlog_Do_DB	
Binlog_Ignore_DB	

- Show connected slaves

Server ID	Host
1	10.23.2.3

Variable	Value
Slave_IO_State	Waiting for master to send event
Master_Host	10.23.2.3
Master_User	admin
Master_Port	3306
Connect_Retry	60
Master_Log_File	mysql-bin.000001
Read_Master_Log_Pos	647
Relay_Log_File	mysqld-relay-bin.000002
Relay_Log_Pos	555
Relay_Master_Log_File	mysql-bin.000001
Slave_IO_Running	Yes
Slave_SQL_Running	Yes

Gambar 4.8 Status *master-slave* pada *raspberry* 2 atau S2

Pada gambar 4.8 dapat diketahui status *master-slave* pada *raspberry* 2 (S2) dimana menjadi master dari IP 10.23.2.3 (S1). Pada gambaran diatas juga diketahui bahwa pada *raspberry* 2 menjadi slave dari IP 10.23.2.3 (S1). Pada tahap ini kedua database sudah terhubung, untuk dapat melihat kedua database terhubung maka dibuat percobaan sederhana yaitu dengan membuat database.

Tabel 4.3 isi pada tabel user ketika semua web server dalam keadaan state atau bekerja.

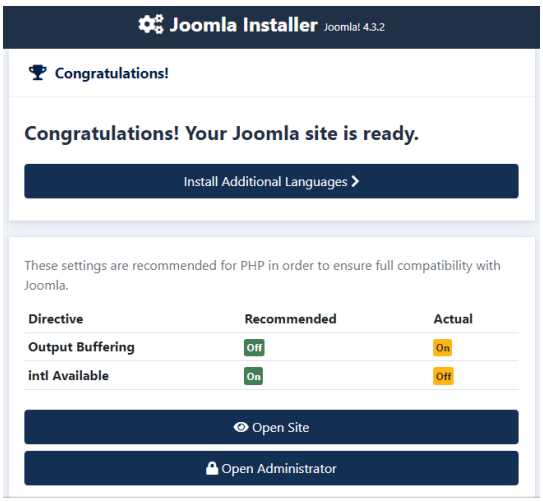
Server	**Isi pada database**

S1 **10.23.2.3**	Database ▲ Collation Master replication Slave replication Action information_schema utf8mb3_general_ci Replicated Replicated Check privileges joomla utf8mb4_general_ci Replicated Replicated Check privileges mysql utf8mb4_general_ci Replicated Replicated Check privileges performance_schema utf8mb3_general_ci Replicated Replicated Check privileges sys utf8mb3_general_ci Replicated Replicated Check privileges
S2 **10.23.2.4**	Database ▲ Collation Master replication Slave replication Action information_schema utf8mb3_general_ci Replicated Replicated Check privileges joomla utf8mb4_general_ci Replicated Replicated Check privileges mysql utf8mb4_general_ci Replicated Replicated Check privileges performance_schema utf8mb3_general_ci Replicated Replicated Check privileges sys utf8mb3_general_ci Replicated Replicated Check privileges

Dari data tabel 4.3 diatas penulis membuat sebuah database joomla pada S1 maka isi dari S2 juga akan mengikuti database dari S1, begitupun pada status *master replication* dan *slave replication* juga berstatus *replicated*. Dengan begitu maka database telah berhsil untuk di replication.

4.6 Instalasi website

Instalasi website ini menggunakan joomla, website ini digunakan sebagai pengujian status loadbalance dan juga sebagai pengujian database. Fungsi dari website ini juga sebagai akomodir client dalam melakukan pemintaan data layaknya server pada umumnya. Pada konfigurasi ini penggunaan database dan konfigurasi loadbalance secara benar sangat penting karena jika ada konfigurasi yang gagal atau tidak tepat maka website yang kita buat tidak muncul interface ataupun mengalami kegagalan dalam membuat isi dari data data website tersebut. Ketika website joomla berhasil terkonfiguras dengan benar maka tampilan awal setelah proses instalasi seperti pada gambar berikut :



Gambar 4.9 Status instalasi pada website joomla

Pada gambar 4.9 instalasi joomla telah berhasil dilakukan, selanjutnya dapat dilakukan pengujian status loadbalance server ketika user melakukan request data dan juga pengujian terhadap create, read, update dan delete pada website yang bertujuan untuk mengetahui apakah data data yang dikirim maupun diterima oleh user sudah sesuai dengan yang di ingkan pada skema kerja bab 3.

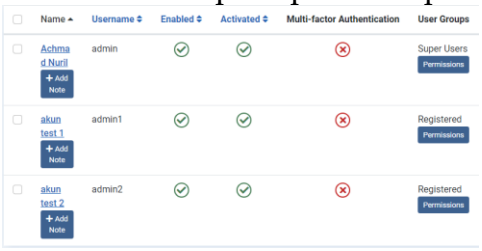
4.7 Pengujian data CRUD

Pengujian ini dilakukan dengan create, read, update dan delete pada website pada bagian admin. Pengujian ini untuk mengetahui apakah ketika salah satu dari webserver loadbalance mengalami downtime otomatis data masih dapat tersimpan pada node yang masih berstatus aktif. Dalam pengujian ini status interface pada webadmin haruslah sama seperti pada database sehingga konfigurasi yang dilakukan dianggap telah sesuai.

Tabel 4.4 isi pada tabel user pada kedua database

Server	Isi pada database
S1 **10.23.2.3**	←T→ id name username email Edit Copy Delete 850 Achmad Nuril admin adminjoomla@gmail.com Edit Copy Delete 851 akun test 1 admin1 akun1@gmail.com
S2 **10.23.2.4**	←T→ id name username email Edit Copy Delete 850 Achmad Nuril admin adminjoomla@gmail.com Edit Copy Delete 851 akun test 1 admin1 akun1@gmail.com

Dari tabel 4.4 dapat diketahui pada kedua databse dalam tabel user memiliki isi yang sama dimana ada 2 user admin. Kemudian pada sisi web server akan ditambahkan lagi sebuah user dan dalam penambahan pada saat itu salah satu server raspberry dimatikan untuk mengetahui apakah data yang tadi dibuat *auto save* pada server yang lainnya atau tidak. Setelah ditemukan data pada database sama, selanjutnya dilakukan membuka website administrator joomla untuk menambahkan user admin2 seperti pada tampilan berikut :



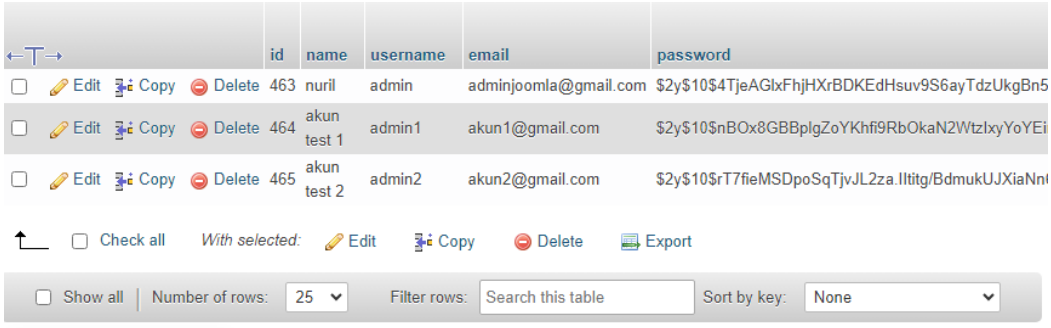
Gambar 4.10 akun administrator

Pada gambar 4.8 diketahui bahwa ada 3 admin, dimana admin yang baru saja ditambahkan adalah admin2. Pada penambahannya salah satu raspberry telah dimatikan untuk menguji server apakah data yang telah dibuat akan tersimpan atau tidak. Setelah itu untuk menguji dengan mengakses halaman database melalui phpmyadmin dan ditapati hasil berikut :

Tabel 4.5 isi pada tabel user ketika salah satu server dimatikan

Server	Isi pada database
S1 **10.23.2.3**	←T→ id name username email Edit Copy Delete 850 Achmad Nuril admin adminjoomla@gmail.com Edit Copy Delete 851 akun test 1 admin1 akun1@gmail.com Edit Copy Delete 852 akun test 2 admin2 akun2@gmail.com
S2 **10.23.2.4**	This site can't be reached 10.23.2.4 took too long to respond. Try: • Checking the connection • Checking the proxy and the firewall • Running Windows Network Diagnostics ERR_CONNECTION_TIMED_OUT

Dari tabel 4.5 tersebut dapat diketahui pada server S1 data yang dituliskan pada web admin dapat disimpan walaupun salah satu server dimatikan, sedangkan pada S2 menunjukkan bahwa tidak dapat menjangkau server. Hal ini sesuai dikarenakan salah satu raspberry yaitu S2 dimatikan. Pengujian selanjutnya adalah dengan menghidupkan kembali server S2, dimana saat S2 hidup data yang tadinya ada pada S1 dan belum ada pada S2 harus *terupdate* dikarenakan adanya replikasi *database.*



Gambar 4.9 isi database S2

Dari gambar 4.9 diatas dapat diketahui bahwa pada server S2 ketika dihidupkan data pada tabel user akan otomatis terupdate sehingga data akan sama

pada *database sever* S1 dan *database server* S2 seperti yang ditunjukkan pada tabel dibawah ini.

Tabel 4.6 isi pada tabel user ketika server S2 dihidupkan kembali

Server	Isi pada database
S1 **10.23.2.3**	Edit Copy Delete 850 Achmad Nuril admin adminjoomla@gmail.com Edit Copy Delete 851 akun test 1 admin1 akun1@gmail.com Edit Copy Delete 852 akun test 2 admin2 akun2@gmail.com
S2 **10.23.2.4**	Edit Copy Delete 850 Achmad Nuril admin adminjoomla@gmail.com Edit Copy Delete 851 akun test 1 admin1 akun1@gmail.com Edit Copy Delete 852 akun test 2 admin2 akun2@gmail.com

Pada tabel 4.6 diatas dapat diketahi bahwa pada server S1 maupun server S2 data yang tadinya ada pada server S1 ketika server S2 mengalami down secara otomatis data akan sama kembali hal ini diarenakan terjadi singkronsasi data ketika server S2 kembali ke keadaan state atau bekerja.

4.8 Pengujian Peformansi

Setelah pengujian database telah selesai selanjutnya adalah pengujian peformansi. Pengujian ini dilakukan untuk mengetahui seberapa peforma yang dapat dilakukan oleh raspberry tersebut. Pengujian ini dengan cara menguji ketika server dilakukan balancing dan ketika server standalone. Pengujian ini menggunakan aplikasi jmeter. Pengujian yang pertama adalah 100 sample dan didapati hasil seperti tabel beriku:

Tabel 4.7 tester balance jmeter 100 samples

Label	# Samples	Average	Min	Max	Std. Dev.	Error %
HTTP Request	100	2617	0	5248	1649.52	0.00%
TOTAL	100	2617	0	5248	1649.52	0.00%

Pada tabel 4.7 *sample* tester yang digunakan adalah 100 *samples* dan dapat diketahi bahwa untuk mencapai 100 *samples* waktu maksimal yang diperlukan adalah 5248ms dengan rara - rata waktu yang dibutuhkan adalah 2617ms. Setelah 100 sample selesai pengujian dilanjutkan dengan 300 sampe sehingga didapati hasil seperti dibawah ini.

Tabel 4.8 tester jmeter 300 samples

Label	# Samples	Average	Min	Max	Std. Dev.	Error %
HTTP Request	300	8377	0	14083	4154.61	0.00%
TOTAL	300	8377	0	14083	4154.61	0.00%

Setelah 100 *sample* tester dilakukan dilanjutkan dengan 300 *samples* dan dapat diketahui pada tabel 4.8 bahwa waktu yang dibutuhkan pada 300 *samples* adalah 14083ms dengan rata rata waktu yang dibutuhkan untuk mengakses adalah 8377ms.

Pengujian ke 3 dengan memasukan sampe untuk percobaan dengan 500 sampe. Data hasil ketika 500 sample diketahui sebagai berikut :

Tabel 4.9 tester jmeter 500 samples

Label	# Samples	Average	Min	Max	Std. Dev.	Error %
HTTP Request	500	12000	0	19747	6203.16	0.00%
TOTAL	500	12000	0	19747	6203.16	0.00%

Pada 500 *sample* di tabel 4.9 dapat diketahui bahwa waktu yang digunakan untuk 500 *sample* maksimal adalah 19747ms dan untuk rata – rata waktu yang diperlukan adalah 12000ms. Pengujian yang terakir dengan memasukkan data sample sebanyak 1000 sample dan dapat diketahui hasilnya seperti tabel dibawah ini.

Tabel 4.10 tester jmeter 1000 samples

Label	# Samples	Average	Min	Max	Std. Dev.	Error %
HTTP Request	1000	19729	0	33749	9682.62	0.00%
TOTAL	1000	19729	0	33749	9682.62	0.00%

Pada sample 1000 seperti tabel 4.110 ketika dilakukan proses terting maka waku maksimal yang dibutuhkan untuk menyelesaikan 1000 request client adalah 33749ms dengan rata-rata waktu adalah 19729ms. Pada semua pengujian menggunakan metode loadbalance tidak terjadi *error* mulai dari 100 hingga 1000 sample.

Setelah pengujian menggunakan metode load balance selesai maka pengujian selanjutnya adalah menguji *raspberry* yang digunakan sebagai server secara individu atau standalone dengan sample data yang digunakan sama seperti pada sample loadbalance. Pertama yang diujikan adalah 100 sample dengan hasil uji seperti tabel yang tertera dibawah ini.

Tabel 4.11 tester standalone 100 samples

Label	# Samples	Average	Min	Max	Std. Dev.	Error %
HTTP Request	100	5585	0	6869	761.74	0.00%
TOTAL	100	5585	0	6869	761.74	0.00%

Dari tabel 4.11 diatas diketahui bahwa pada saat raspberry server di berikan data sample 100 maka waktu maksimal yang dibutuhkan adalah 6869ms dimana dengan 100 sample tersebut rata-rata waktu yang dibutuhkan 5585ms. Setelah hasil dari 100 sample diketahui dilanjutkan dengan pengujian ke 2 yaitu dengan jumlah sample yang di ujikan sebanyak 300 sample. Dari 300 sample tersebut didapati hasil sebagai berikut :

Tabel 4.12 tester standalone 300 samples

Label	# Samples	Average	Min	Max	Std. Dev.	Error %
HTTP Request	300	12948	0	21096	6777.47	0.00%
TOTAL	300	12948	0	21096	6777.47	0.00%

Pada 300 sample data yang telah ditest pada raspberry server, dapat diketahui bahwa pada tabel 4.12 waktu yang dibutuhkan ketika 300 sample di coba adalah 21096ms dengan rata-rata waktu yang dibutuhkan 12948ms. Setelah pengujian sample 300 selesai dilanjutkan dengan pengujian sample sebanyak 500 sample. Dalam pengujian ini hasil yang diberikan pada aplikasi jmeter dapat diketahui sebagai berikut :

Tabel 4.13 tester standalone 500 samples

Label	# Samples	Average	Min	Max	Std. Dev.	Error %
HTTP Request	500	20540	0	33377	10001.06	0.00%
TOTAL	500	20540	0	33377	10001.06	0.00%

Selanjutnya terpadat pengujian dengan sample 500 ditunjukkan dengan tabel 4.13. dari gambar tersebut dapat diketahui bahwa waktu untuk 500 sample maksimal 33377ms dengan waktu rata – rata yang dibutuhkan 20540ms. Untuk pengujian yang terakir data sample yang diberikan adalah 1000 sample. Hal ini sama ketika raspberry digunakan dalam sistem load balance. Untuk hasil data 1000 sample ketika raspberry standalone dapat diketahui dari tabel berikut ini.

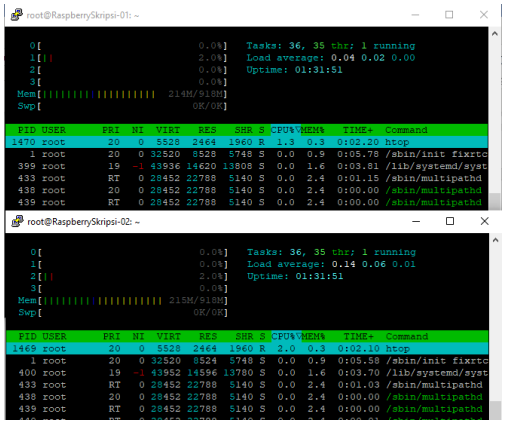
Tabel 4.14 tester standalone 1000 samples

Label	# Samples	Average	Min	Max	Std. Dev.	Error %
HTTP Request	1000	36281	0	64413	18840.83	4.20%
TOTAL	1000	36281	0	64413	18840.83	4.20%

Pada percobaan terakir data sample yang digunakan adalah 1000 sample dimana dari data sample 1000 tersebut waktu maksimal yang digunakan untuk menyelesaikan adalah 64413ms dengan rata – rata waktu 36281ms yang ditunjukkan pada tabel 4.15.

4.9 Peformansi user riil

Setelah pengujian peformansi dilakukan menggunakan Jmeter dan dengan user sampling secara virtual, pengujian akan dilanjutkan dengan pengujian server dengan beban riil dimana beban uji yang diberian adalah 6 user dimana dari 6 user tersebut melakukan akses website secara bersamaan kemudian pada server S1 dan server S2 telah dibuka aplikasi untuk melihat kinerja dari perangkat raspberry tersebut sehingga akan dapat terlihat beban kerja pada server yang diujikan. Dari hasil uji tersebut dapat diketahui ketika akses user riil tersebut kinerja server sebagai berikut ini :



Gambar 4.11 peforma dari raspberry web server

Pada gambar 4.9 diatas dapat diketahui ketika pengujian secara riil dan diakses secara bersamaan baik server 1 maupun server 2 hanya bekerja 2% pada salah satu core prosesornya dan kinerja ram sebesar 214-215M. sehingga dapat diketahui untuk 6 user tersebut sangat ringan untuk ditangani oleh web server raspberry.

5.1Kesimpulan

Berdasarkan ujicoba penulis dapat menyimpulkan sebagai berikut :

1. Dalam pengujian yang dilakukan 2 raspberry pi yang digabungan menggunakan loadbalance ketika mengakses 1000 user secara virutal tidak ditemukan *error* dan tidak terjadi *down* pada ke 2 server web raspberry pi.
2. Pada pengujian menggunakan user riil sebanya 6 user yang terkoneksi secara wireless dan mengakses website secara bersamaan, ke-2 web server dalam keadaan baik dan tidak terjadi lonjakan kinerja dimana hanya terdapat 2% kineja pada salah satu core prosesornya dan penggunaan ram hanya sebesar 214-215M dari 1GB yang dimiliki hardware raspberry pi.
3. Dari hasil testing pada pengujian ini, raspberry pi dapat dijadikan sebagai clustering server dengan hasil lebih cepat yaitu dengan waktu 3379 ms ketika mengakses 1000 *samples* dibandingkan ketika raspberry berdiri sendiri atau *stand alone* dengan waktu akses 1000 *samples* selama 64413ms.

DAFTAR PUSTAKA

[1] P. A. Aditya, M. A. Irwansyah dan H. Priyanto, " Rancang Bangun Web Server Berbasis Linux Dengan Metode Load Balancing (Study Kasus : Laboratorium Teknik Informatika) " Jurnal Sistem dan Teknologi Informasi (*JUSTIN*) *Vol. 3, No. 1*, (2016).

[2] M. A. Nugroho, M. kom. dan R. Katardi, " Analisis Kinerja Penerapan Container Untuk Load Balancing Web Server Pada Raspberry Pi " JIPI Volume 01, Nomor 02, Desember : 7 – 15.

[3] R. H. Putra dan W. Sugeng, "Implementasi Cluster Server pada Raspberry Pi dengan Menggunakan Metode Load Balancing," Jurnal Edukasi dan Penelitian Informatika (JEPIN) Vol. 2, No. 1, vol. 2, 2016.

[4] R. Dawood,S. F. Qiana dan S. Muchallil, "Kelayakan Raspberry Pi sebagai Web Server: Perbandingan Kinerja Nginx, Apache, dan Lighttpd pada Platform Raspberry Pi," *Jurnal Rekayasa Elektrika Vol. 11, No. 1, April 2014, hal. 25-29.*

[5] D.T Nugrahadi, R. Herteno dan M. Anshari "Pengaruh Implementasi Load Balancing dan Tuning Web Server Pada Response Time Raspberry Pi" Kumpulan jurnaL Ilmu Komputer (KLIK) Volume 06, No.02 Juni 2019 ISSN: 2406-7857.

[6] I. D. Wijaya, U. Nurhasan, M. Agung Barata, "IMPLEMENTASI RASPBERRY PI UNTUK RANCANG BANGUN SISTEM KEAMANAN PINTU RUANG SERVER DENGAN PENGENALAN WAJAH MENGGUNAKAN METODE *TRIANGLE FACE" Jurnal Informatika Polinema* ISSN: 2407-070X Volume 4, Edisi 1, November 2017.

[7] Much. Aziz Muslim, "Pengembangan Distro Ubuntu untuk Aplikasi game Centre" *Jurnal Teknologi Informasi DINAMIK Volume XI, No.1 Januari 2006 :16-22.*

[8] Wibowo, Ari Setyo " File share menggunakan Turnkey Owncloud" *database karya ilmiah civitas akademika UDI, April 2018.*

[9] Suryanto, "IMPLEMENTASI CLUSTERING DATABASE SERVER MENGGUNAKAN PGCLUSTER UNTUK OPTIMALISASI KINERJA SISTEM BASIS DATA" *Jurnal Teknik Komputer Amik BSI Vol.1 No. 1 Februari 2015.*

[10] Toni Bourke, "Server Load Balancing 1st edition." [Online]. Available: books.google.co.id

[11] Hexa Time "DevOps Expert Haproxy – High Peformance Load Balancer Server" 2019.

[12] Faris M. A, M. Data, Heru N. "Perbandingan Kinerja *Haproxy dan Zevenet* Dalam Pengimplementasian *Multi Service Load Balancing*" *Jurnal Pembangunan Teknologi Informasi dan Ilmu Kompuer Vol 3, No.1 Januali 2019. Hlm 253-260*

[13] Kiki Yuniar, H. Endah, M. Idhom "IMPLEMENTASI REVERSE PROXY PADA HOSTING WEB SERVER DI DOCKER CONTAINER" *Jurnal Informatika dan Sistem Informasi (JIFoSI) Vol.3, No.1. April 2022.*