Komputika: Jurnal Sistem Komputer
Vol. 8, No. 1, April 2019, hlm. 37 - 43
DOI: 10.34010/komputika.v8i1.1623

p-ISSN: 2252-9039
e-ISSN: 2655-3198

Implementasi Cluster Server Pada Raspberry Pi dengan Menggunakan Metode Load Balancing

Implementation of the Cluster Server on the Raspberry Pi Using Load Balancing

R A Putra[1], A P Sujana[2*]

[1,2)]Program Studi Sistem Komputer, Fakultas Teknik dan Ilmu Komputer, Universitas Komputer Indonesia
Jl. Dipati Ukur No. 112 – 116, Bandung, Indonesia 40132
aprianti.putri.sujana@email.unikom.ac.id

ABSTRACT *– When the server is overloaded, this will make the server slow down, the effect is that the service access time increases which will make the client uncomfortable in using the services that are on the server. Cluster servers offer a solution so that the load is divided evenly, this system utilizes several computers that are configured in such a way that they can work together. The results of cluster server testing show an increase in http request response time by 59% faster than without using a server cluster system. Another advantage of using a cluster server is the existence of redundancy, when one node member is inactive the master node or load balancer will automatically move the request to the active node.*

Keywords *– Raspberry Pi, Server, Cluster, Load Balancing.*

ABSTRAK *– Ketika server mengalami overload, hal ini akan membuat server melambat, efeknya adalah waktu akses layanan bertambah yang akan membuat client tidak nyaman dalam menggunakan layanan yang ada pada server. Cluster server menawarkan solusi agar beban terbagi secara merata, sistem ini memanfaatkan beberapa komputer yang di konfigurasi sedemikian rupa agar dapat berkerja bersama-sama. Hasil dari pengujian cluster server menunjukan peningkatan waktu respon http request sebesar 59% lebih cepat dibandingkan tanpa menggunakan sistem cluster server. Keuntungan lain menggunakan cluster server adalah adanya redudancy, jadi ketika salah satu anggota node tidak aktif maka secara otomatis master node atau load balancer akan memindahkan request menuju node yang masih aktif.*

Kata Kunci *– Raspberry Pi, Server, Cluster, Load Balancing.*

1. PENDAHULUAN

Ketika kita mengakses internet misalnya untuk membuka email, mengakses konten multimedia atau sekedar browsing membaca berita, di balik semua itu ada sebuh komputer yang melayani permintaan kita sebagai client agar apa yang kita inginkan dapat terpenuhi, komputer sering disebut server. Tentu sebagai client menginginkan ketika kita meinta sesuatu server dapat mengerjakanya secapat mungkin agar kita tidak harus menunggu, tetapi ketika permintaan yang masuk terlalu banyak server akan mengalami *overload* yang berakibat bertambahnya waktu akses menuju server. Tentu hal ini sangat tidak di inginkan untuk terjadi, ini menjadi tugas seorang administrator untuk mencari bagaimana caranya agar hal tersebut tidak terjadi.

Sistem *cluster server* bisa menjadi salah satu solusi agar hal tersebut tidak terjadi, yang di maksud dengan *cluster server* adalah gabungan dari beberapa komputer sekaligus yang telah di konfigurasi agar dapat berkerja bersama-sama. Sistem ini menawarkan peningkatan performa serta menyediakan *redudancy* fungsinya, ketika salah satu server mengalami gangguan secara otomatis server lain akan mengambil alih tugas nya, sehingga disisi *client* tidak akan terlihat meskipun beberapa server sedang mengalami gangguan.

Copyright © 2019, Komputika: Jurnal Sistem Komputer, p-ISSN: 2252-9039, e-ISSN: 2655-3198

2. METODE DAN BAHAN

A. *Server*

Server, sesuai dengan namanya bisa diartikan sebagai pelayan pada suatu jaringan komputer. Server adalah komputer yang berfungsi untuk melayani, membatasi, dan mengontrol akses terhadap user-user dan sumber daya pada suatu jaringan komputer. Server merupakan suatu bagian terpenting dari sebuah jaringan. suatu jaringan komputer dengan banyak komputer memerlukan suatu server yang bertugas untuk menyediakan layanan yang dibutuhkan oleh user. [1]

Server memiliki beberapa macam jenis. Dari berbagai macam jenis server tersebut memiliki kegunaan dan tugasnya masing – masing. Dan setiap jenis server tersebut bisa dijadikan satu mesin. Jadi satu komputer server bisa melayani berbagai service atau layanan dan juga bisa menggunakan sistem *cluster server* yang membagi setiap layanan pada *node-node* yang tersedia. [2]

B. *Web Server*

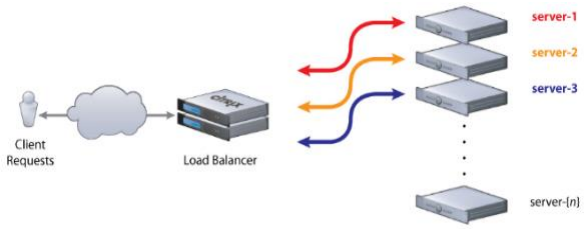
Web Server merupakan perangkat lunak yang memiliki fungsi menerima permintaan berupa halaman web melalui protokol *HTTP* atau *HTTPS* dari suatu klien yang lebih dikenal dengan nama browser, kemudian mengirimkan kembali dalam bentuk halaman-halaman web yang umumnya berbentuk dokumen HTML. Fungsi utama sebuah server web adalah untuk mentransfer berkas atas permintaan pengguna melalui protokol komunikasi yang telah ditentukan. Disebabkan sebuah halaman web dapat terdiri atas berkas teks, gambar, video, dan lainnya pemanfaatan server web berfungsi pula untuk mentransfer seluruh aspek pemberkasan dalam sebuah halaman web yang terkait termasuk di dalamnya teks, gambar, video, atau lainnya. Penggunaan biasanya melalui aplikasi pengguna seperti peramban web, meminta layanan atas berkas ataupun halaman web yang terdapat pada sebuah server web, kemudian server sebagai manajer layanan tersebut akan merespon balik dengan mengirimkan halaman dan berkas-berkas pendukung yang dibutuhkan, atau menolak permintaan tersebut jika halaman yang diminta tidak tersedia. Saat ini pada umumnya server web telah dilengkapi pula dengan mesin penerjemah bahasa skrip yang memungkinkan server web menyediakan layanan situs web dinamis dengan memanfaatkan pustaka tambahan seperti PHP, dan ASP. [3]

C. *HTTP (Hyper Text Transfer Protocol)*

Protokol yang dipergunakan untuk mentransfer dokumen dan web dalam sebuah web browser, melalui www. HTTP juga merupakan protokol yang meminta dan menjawab antar klien dan server. [4]

D. *Load Balancing*

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. [3]



Gambar 1. Ilustrasi cara kerja load balancing

E. *QOS*

Quality of Service atau kualitas layanan adalah sebuah standar untuk memastikan bahwa suatu performa pada suatu jaringan berjalan dengan baik. Tujuan *QOS* adalah untuk menghemat penggunaan *bandwith*, mengurangi kemungkinan *dataloss* dan untuk mengendalikan *latency* serta *jitter*. [5] [6]

F. *Raspberry Pi*

Raspberry Pi adalah sebuah *single board computer* (mini PC) pertama kali diluncurkan pada tahun 2012. Raspberry Pi dibuat/dikembangkan oleh sebuah perusahaan yang bernama Raspberry Pi Foundation dari UK. Perusahaan ini membuat raspberry ri bertujuan untuk mempromosikan pengajaran ilmu komputer di sekolah dasar. Raspberry Pi menggunakan system on a chip (SoC) dari Broadcom BCM2837 dan tidak menggunakan hard disk, namun menggunakan SD Card untuk proses booting dan penyimpanan data jangka panjang.

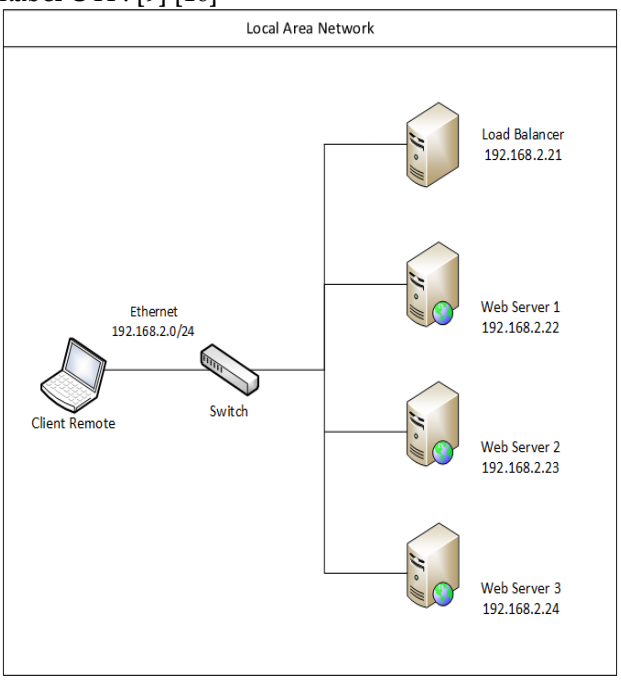
G. *Ansible*

Ansible adalah software otomatisasi *open source* yang mengotomatisasi penyediaan perangkat lunak, manajemen konfigurasi, dan pemasangan aplikasi. Ansible disertakan sebagai bagian dari distro Fedora Linux, yang dimiliki oleh Red Hat Inc., dan juga tersedia untuk Red Hat Enterprise Linux, CentOS, dan Scientific Linux melalui Paket Ekstra untuk

Enterprise Linux (EPEL) dan juga untuk sistem operasi lain. Seperti kebanyakan manajemen konfigurasi perangkat lunak, *Ansible* memiliki dua tipe server: mesin pengontrol dan *node*. Pertama, terdapat satu mesin pengontrol tunggal dimana mesin ini akan melakukan orkestrasi atau mengatur setiap *node* yang ada. *Node* dikelola oleh mesin pengontrol melalui SSH. Mesin pengontrol memetakan lokasi dari setiap node melalui "*inventory*". Untuk mengatur *node*, *Ansible* menyebarkan modul-modul ke *node* yang ada melalui SSH. Modul-modul ini disimpan sementara di setiap *node* dan berkomunikasi dengan mesin pengontrol melalui protokol JSON. Ketika *Ansible* tidak sedang mengelola *node*, ia tidak mengkonsumsi sumber daya karena tidak ada *daemon* atau program yang dijalankan untuk *Ansible* di latar belakang. Berbeda dengan manajemen konfigurasi perangkat lunak populer - seperti *Chef*, *Puppet*, dan *CFEngine* - *Ansible* menggunakan arsitektur tanpa agen. Dengan arsitektur berbasis agen, *node* harus memiliki *daemon* terpasang secara lokal yang berkomunikasi dengan mesin pengendali. Dengan arsitektur tanpa agen, *node* tidak diharuskan memasang dan menjalankan *daemon* untuk terhubung dengan mesin jaringan dengan mencegah *node* untuk terus mengecek apakah mesin pengendali telah siap atau tidak. [7] [8]

H. *Perancangan*

Secara umum topologi yang digunakan untuk membangun *cluster server* adalah topologi star, dimana setiap raspberry pi dan *client* akan dihubungkan dengan sebuah *switch* dengan media kabel UTP. [9] [10]

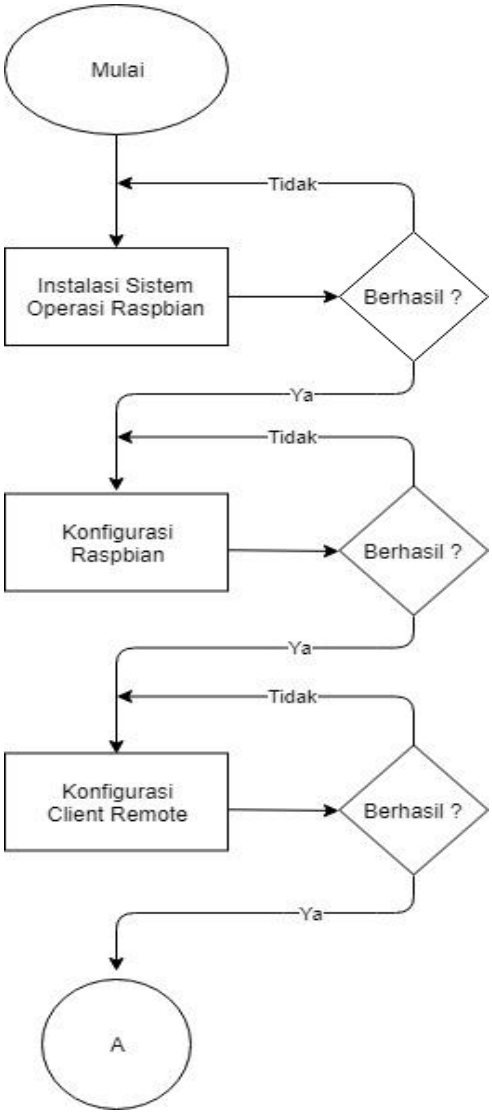


Gambar 2. Topologi Logika Raspberry Pi Cluster

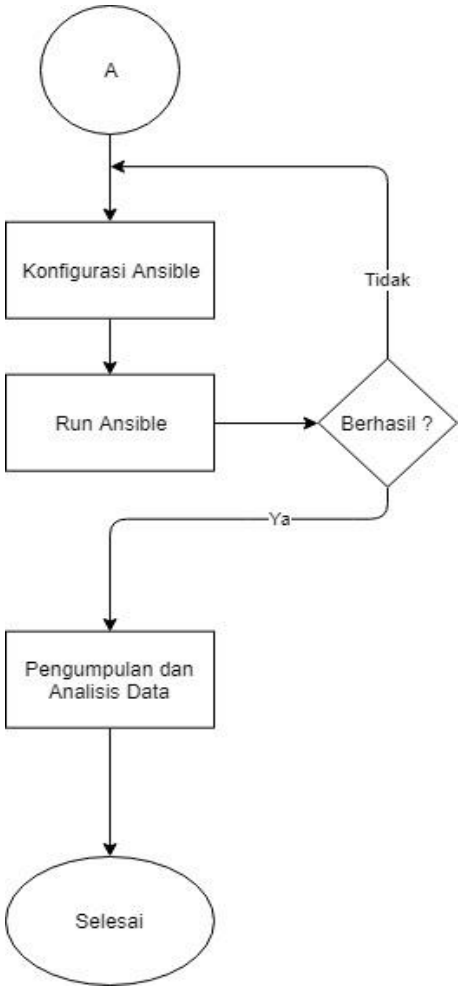
Berdasarkan pada gambar 2 perancangan sistem ini terdiri dari 4 buah raspberry pi, 1 berfungsi sebagai *load balancer* dan 3 berfungsi sebagai *web server*. Terdapat *switch* yang berfungsi untuk menghubungkan semua raspberry pi agar dapat saling berkomunikasi, semua perangkat berada pada network 192.168.2.0 dengan subnet 255.255.255.0.

I. *Instalasi dan Konfigurasi Sistem*

Pada bagian ini menjelaskan tentang bagaimana perangkat lunak di konfigurasi untuk membangun sebuah *cluster server*. Gambar 3 dan 4 merupakan diagram alir tahapan-tahapan pada perancangan sistem.



Gambar 3. Diagram alir perancangan sistem 1



Gambar 4. Diagram alir perancangan sistem 2

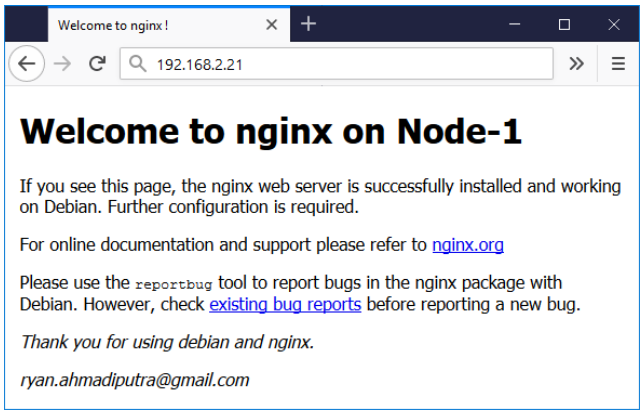
Pada saat instalasi *raspbian* selesai ada proses yang harus dilakukan terlebih dahulu sebelum sdcard di masukan kedalam raspbery, yaitu proses menyalakan *service ssh* yang bertujuan ketika raspberry pertama kali *booting service ssh* telah aktif, sehingga tidak diperlukan monitor, mouse dan keyboard saat konfigurasi, seluruh konfigurasi dilakukan via ssh. Ada beberapa hal yang perlu di ingat sebelum menggunakan *ansible*. *Ansible* memerlukan *password less* ssh maksudnya adalah ketika ingin menggunakan *ansible*, *client remote* harus dapat login pada setiap raspberry tanpa menggunakan *password* tetapi menggunakan *ssh keys*, yang harus di daftarkan terlebih dahulu, pada setiap raspberry yang akan menjadi target konfigurasi *ansible*.

3. HASIL DAN PEMBAHASAN

A. *Pengujian*

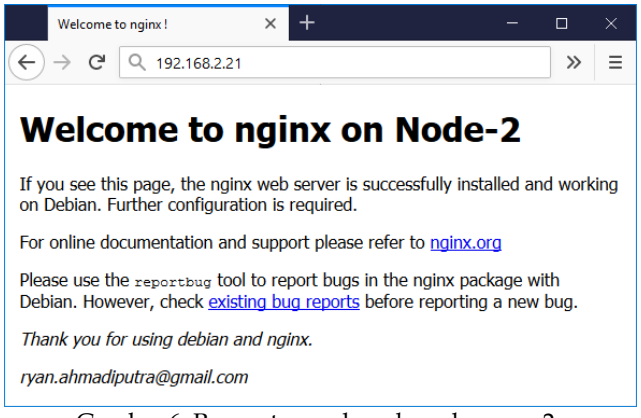
Pengujian dilakukan untuk mengatahui kelebihan dan kekurangan ketika menggunakan sistem *cluster server* serta melihat perbandingan performa algoritma *round robin* dan *ip hash* yang digunakan pada *load balancer*.

B. *Uji Akses cluster server*



Gambar 5. Request masuk pada web server 1

Ketika ip *load balancer* di akses sebuah *static page* seperti pada gambar 5 akan muncul, dan seperti disebutkan *request* tersebut telah diproses oleh web server nomor satu dan ketika halaman di *refresh* maka *request* ke dua akan di proses oleh server nomor dua seperti yang di tunjukan oleh gambar 6, dan ketika di *refresh* lagi maka *request* akan di proses oleh server nomor 3 dan *request* selanjutnya akan kembali lagi di proses oleh server nomor satu dan seterusnya berulang. Dapat di simpulkan bahwa algoritma yang digunakan adalah *round robin*. Karena dengan algoritma tersebut *load balancer* akan mendistribusikan *request* berdasarkan waktu kedatangan. Sedangkan ketika menggunakan algoritma *ip hash request* akan di distribusikan berdasarkan alamat *ip client*. Ketika *request* pertama di proses oleh web server nomor 2 maka *request-request* selanjutnya dari sumber ip yang sama akan di proses oleh web server yang sama.



Gambar 6. Request masuk pada web server 2

C. *Pengujian menggunakan iperf*

Pengujian ini bertujuan untuk mengecek seberapa besar *bandwidth* yang dapat dikirim dan diterima antara raspberry dan client remote. Pengujian dilakukan dengan menggunakan software yang bernama *iperf* yang umum digunakan untuk

mengukur maximum *bandwidth* yang dapat di lewatkan pada sebuah jaringan. Topologi pengujian masih sama seperti yang ditunjukan oleh gambar 2. Berikut perintah yang digunakan dalam pengujian menggunakan iperf.

```
$ iperf -c 192.168.2.21 -r
```

Dari perintah tersebut didapatkan hasil seperti berikut.

Tabel 1. Hasil pengujian iperf.

Server	Client	Bandwidth
Raspberry	Client Remote	94.2 Mbits/sec
Client Remote	Raspberry	94.1 Mbits/sec

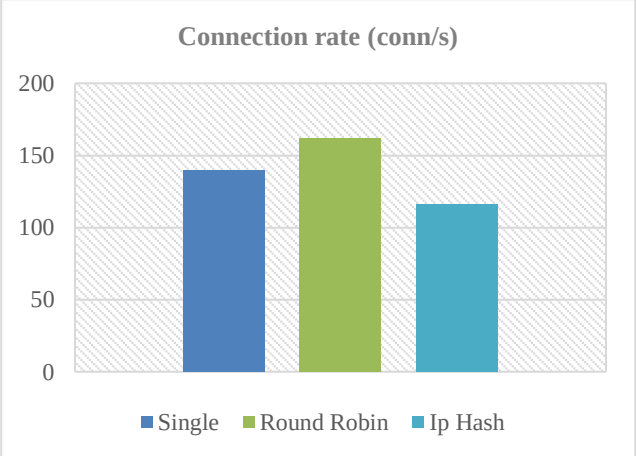
Dapat di lihat bahwa bandwidth yang tersedia antara *client remote* dan raspberry berkisar pada 94Mbps yang artinya *bandwidth* yang tersedia telah sesuai dengan spesifikasi raspberry yang masih menggunakan ethernet dengan kecepatan maximal 100Mbps.

D. Pengujian menggunakan httperf

Pengujian menggunakan software *httperf* dilakukan untuk melihat kemampuan *cluster server* dalam menangani *request*, *httperf* adalah sebuah software *load generator* yang dapat digunakan untuk menguji kemampuan sebuah *webserver*. Berikut perintah *httperf* yang digunakan saat proses pengujian.

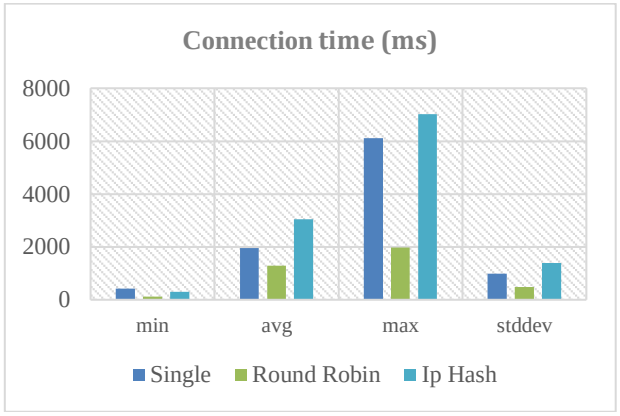
```
$ httperf --server www.ryan.co.id
--num-conn 1000 --num-calls=15 --
rate=200 --timeout 10
```

Perintah di atas akan membuat 1000 koneksi TCP dengan *rate* 200 koneksi per detik serta total 15000 *http request* dengan *rate* 3000 *http request* per detik. Berikut hasil pengujian menggunakan httperf.



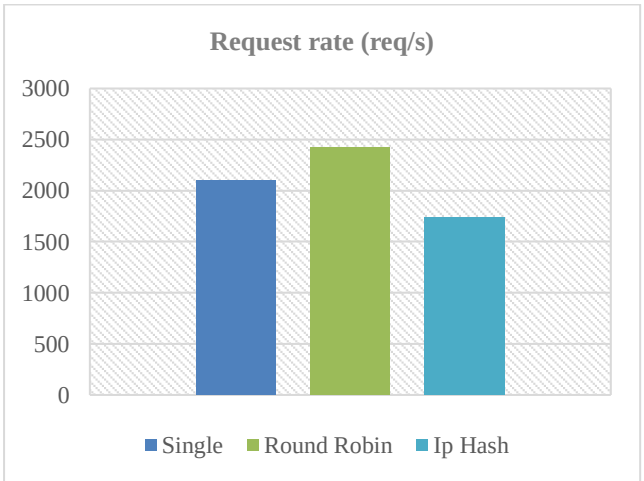
Gambar 7. Connection rate

Ketika menggunakan algoritma *round robin*, server dapat melayai 161 koneksi TCP per detik, jika dibandingkan dengan *ip hash* (116 conn/s), menggunakan algoritma *round robin* pada *load balaner* dapat melayani 39% lebih banyak koneksi TCP, Dan ketika dibandingkan tanpa *load balancing*, terlihat bahwa menggunakan *load balancing* dapat meningkatkan *connection rate* sebesar 16%.



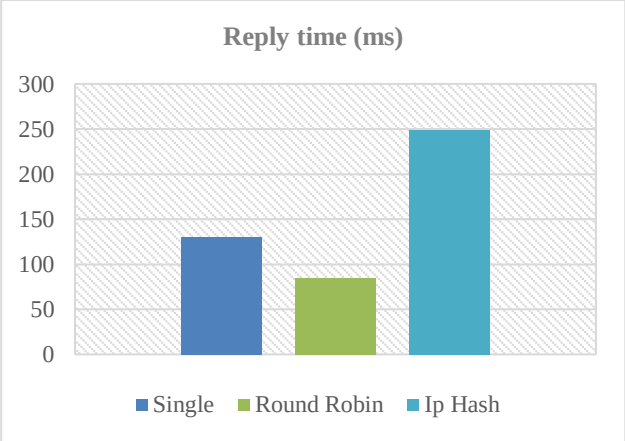
Gambar 8. Connection time

Semakin rendah nilai *connection time* berarti server lebih cepat dalam mengerjakan permintaan koneksi TCP yang di kirim oleh *client*. Data di atas adalah data waktu berapa lama server merespons setiap permintaan koneksi TCP yang di minta oleh client. Waktu di hitung dari mulai *client* meminta, hingga koneksi TCP di tutup. Jika membahas analisa sebelumnya (*connection rate*) menggunakan *load balancer* hanya memiliki selisih 16% di bandingkan dengan *single* raspbery, tetapi dapat di lihat ketika menggunakan *load balancer* (algoritma *round robin*) server dapat menyelesaikan setiap permintaan koneksi TCP lebih cepat meskipun ketika menggunakan algoritma *ip hash* server justru membutuhkan waktu yang sedikit lebih lama.



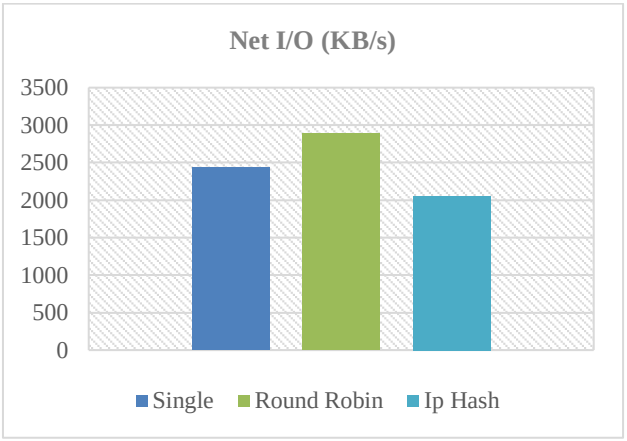
Gambar 9. Request rate

Semakin tinggi nilai *request rate* maka kemampuan server dalam menangani *HTTP request* semakin baik. Pada saat pengujian berlangsung konten web server hanya berupa *static page* yang berupa teks HTML sederhana tanpa konten gambar, css ataupun js. Maka hanya di butuhkan satu *HTTP request* saja untuk memuat seluruh halaman. Ketika website berisi konten-konten di namis maka dibutuhkan beberapa *HTTP request* agar website termuat seluruhnya.



Gambar 10. Reply time

Reply time yang di maksud adalah waktu respons server terhadap permintaaan *HTTP request* yang di kirim oleh client, semakin rendah nilainya maka server dapat melayani *HTTP request* lebih cepat, waktu di hitung sejak server menerima *request* hingga server mengirimkan *reply*. *Reply time* masuk dalam bagian *request section* sehingga masih berhubungan dengan *request rate*. Data yang tertera adalah data waktu rata-rata yang dibutuhkan server untuk membalas permintaan client. Terlihat bahwa ketika menggunakan algoritma *round robin* server dapat membalas permintaan client 59% lebih cepat dibandingkan dengan tanpa menggunakan *load balancer*.

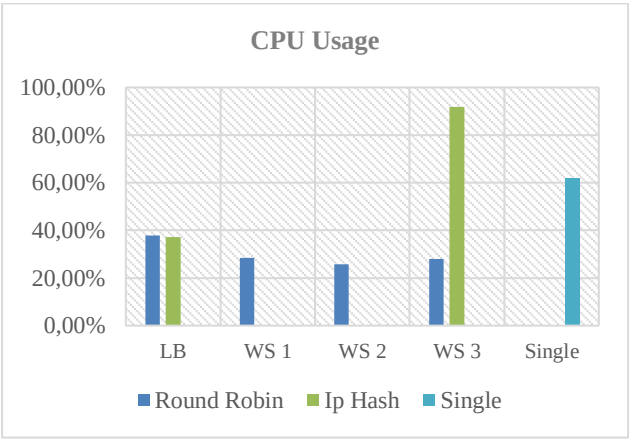


Gambar 11. Net I/O

Net I/O atau dapat disebut juga *throuhput*. Dengan setting dan pengujian yang sama menggunakan *load balancer* (algoritma *round robin*) menghasilakan *throughput* 18% lebih besar di banding tanpa menggunakan *load balancer*.

E. Pengujian menggunakan htop

Pengujian menggunakan *htop* dilakukan untuk melihat seberapa besar beban kerja server ketika menggunakan algoritma *round robin* dan *ip hash*.



Gambar 12. CPU usage

Agar mempermudah analisa data diatas adalah hasil kalkulasi nilai rata-rata untuk seluruh *core*. Sebagai catatan grafik yang paling kanan adalah *single* raspberry atau tanpa menggunakan *load balancer*. Karena algoritma *round robin* membagi request berdasarkan waktu kedatanganya sehingga dapat dilihat bahwa *request* terdistribusi secara merata pada seluruh *node* web server. Sedangkan ketika menggunakan algoritma *ip hash*, karena algoritmanya berdasarkan *ip address client* maka ketika dilakukan pengujian menggunakan *httperf request* hanya di kirim pada web server ke-3 saja, karena ip *client remote* hanya satu tidak berganti secara otomatis seuruh *request* di kirim pada server yang sama.

4. KESIMPULAN

Dari hasil pengujian dapat di lihat bahwa hampir di semua parameter uji ketika menggunakan *load balancer* terdapat peningkatan komampuan server dalam menangani request baik disisi jumlah ataupun waktu. Kelebihan lainya ketika menggunakan *custer server* adalah adanya *redundancy*, sehingga ketika salah satu web server mengalami gangguan secara otomatis *load balancer* akan mengalihkan *request* menuju server yang masih aktif. Dengan demikian *client* masih dapat mengakses website meskipun beberapa server nya sedang tidak aktif. banyak cara yang dapat dipakai untuk membangun *cluster server*, penggunaan *cluster server* harus di seuaikan dengan tujuan penggunaan dan keinginan pengguna.

Penggunaan *ansible* dalam proses pembangunan sistem *cluster server* adalah salah satu cara yang dapat digunakan untuk mempermudah proses pembangunan sistem. Dengan menggunakan bantuan *ansible* kita dapat dengan mudah menambah jumlah *node* atau mengkonfigurasi seluruh *node* tanpa melakukan konfigurasi yang berluang-ulang untuk setiap *node*.

DAFTAR PUSTAKA

[1] SMK Negeri 7 Semarang, "Membangun Server Virtualisasi Aplikasi Menggunakan Citrix XenApp 6.5," 2017.

[2] F. D. Anggraeni, "Internet, Server, Protokol, And Search Engine," 2013.

[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] A. P. Sujana, "Perangkat Pendukung Forensik Lalu Lintas Jaringan," *Jurnal Teknik Komputer Unikom - Komputika,* vol. 3, pp. 31-37, 2014.

[5] B. A. Forouzan, Data Communications and Networking, New York: McGraw Hill, 2007.

[6] B. A. Forouzan, TCP/IP Protocol Suite (Vol 4), New York: Mc Graw-Hill, 2007.

[7] F. Rozak, "Implementasi Monitoring Jaringan Menggunakan Protokol SNMP Pada Mini PC Cubieboard," 2014.

[8] A. Nugroho, W. Yahya dan K. Amron, "Analisis Perbandingan Performa Algoritma Round Robin Dan Least Connection Untuk Load Balancing Pada Software Defined Network," *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer,* vol. 1, no. 12, pp. 1568-1577, 2017.

[9] M. Rosalia, R. Munadi dan R. Mayasari, "Implementasi High Availability Server Menggunakan Metode Load Balancing Dan Failover Pada Virtual Web Server Cluster," *e-Proceeding of Engineering,* vol. 3, no. 3, p. 4496, 2016.

[10] S.-J. Jung, Y.-M. Bae dan W. Soh, "Web Performance Analysis of Open Source Server Virtualization Techniques," *International Journal of Multimedia and Ubiquitous Engineering,* vol. 6, 2011.

[11] D. Fablius, "Rancang Bangun Sistem Penentuan Keputusan Untuk Distribusi Penyedian Kontainer Dengan Multi Kriteria Secara Dinamis," 2017.