

IMPLEMENTASI HIGH-AVAILABILITY WEB SERVER MENGGUNAKAN LOAD BALANCING AS A SERVICE PADA OPENSTACK CLOUD

IMPLEMENTATION HIGH-AVAILABILITY WEB SERVER USING LOAD BALANCING AS A SERVICE ON OPENSTACK CLOUD

Ivan Hidayah¹, Dr. Ir. Rendy Munadi, M.T.², Indrarini Dyah Irawati, S.T., M.T.³

^{1,2,3}Prodi S1 Teknik Telekomunikasi, Fakultas Teknik Elektro, Universitas Telkom

¹ivanhidayah90@gmail.com, ²Rendymunadi@telkomuniversity.ac.id,

³Indrarini@tass.telkomuniversity.ac.id

Abstrak

Penerapan *load balancing* saat ini dirasa kurang efektif dalam penerapan *cloud computing* karena penyedia layanan harus menyediakan server khusus untuk penerapan *load balancing* tersebut. Penerapan *load balancer* di dalam *cloud* dapat menjadi lebih efektif adalah dengan menerapkan layanan *load balancer* di dalam *cloud* yaitu LBaaS. Layanan *load balancer* di dalam OpenStack ini dapat menyediakan algoritma *load balancing round robin*, *least connection*, dan *source ip* secara efektif di dalam *cloud*.

Pada penelitian ini telah dilakukan implementasi *high-availability web server* dengan menggunakan layanan *load balancer as a service* pada OpenStack. Dari hasil penelitian yang dilakukan diketahui bahwa kinerja server yang menggunakan *load balancing* lebih baik dibandingkan dengan *single server*. Hal ini ditunjukkan dengan peningkatan nilai *throughput* sebesar 65,61%, penurunan nilai *elapsed time* sebesar 46,93%, penurunan nilai *response time* sebesar 46,87%, peningkatan nilai *transaction rate* sebesar 66,97%, dan penurunan nilai *cpu utilization* sebesar 58,74%. Dari parameter *fairness*, algoritma *round robin* lebih *fair* dengan nilai *fairness index* mencapai 1 jika dibandingkan dengan algoritma *least connection*.

Kata Kunci: *cloud computing, openstack, load balancer, lbaas, web server*

Abstract

The implementation of *load balancing* is currently considered less effective in the implementation of *cloud computing* because service providers must provide a server for the implementation of *load balancing*. The application of *load balancers* in the *cloud* can be more effective is to implement *load balancer services* in the *cloud*. The service is contained in OpenStack called *Load Balancing as a Service (LBaaS)*. This *load balancer service* in OpenStack can provide *round robin load balancing algorithms*, *least connections*, and *source ip* effectively in the *cloud*.

In this research, a *high-availability web server* has been implemented using *load balancer as a service* on OpenStack. From the results of research it is known that the performance of servers that use *load balancing* is better than that of a *single server*. Proven with increase *throughput* of 65,61%, decrease *elapsed time* of 46,93%, decrease *response time* of 46,87%, increase *transaction rate* of 66,97%, and decrease *cpu utilization* of 58,74%. But from the *fairness* parameter, the *round robin algorithm* is more fair with a *fairness index* value of 1 when compared to the *least connection algorithm*.

Keywords: *cloud computing, openstack, load balancer, lbaas, web server*

1. Pendahuluan

Seiring berkembangnya bisnis *platform cloud* saat ini, memberikan layanan yang lebih cepat dan lebih handal telah menjadi masalah yang perlu dipecahkan segera [1]. Meningkatnya permintaan akan kebutuhan informasi dalam internet menyebabkan trafik dalam internet juga semakin padat yang mengakibatkan beban kerja pada suatu penyedia *web server* akan mengalami kelebihan beban, sehingga dapat menyebabkan server tersebut tidak bisa diakses.

Salah satu cara untuk mengatasi *overload* dalam server *cloud* dapat diterapkan penggunaan *load balancer*. *Load balancing* bertujuan untuk memaksimalkan *throughput*, menghindari *overload*, mengurangi konsumsi energi dengan cara membagi beban *request*, meminimalkan *response time*, dan mengurangi *network latency* [2]. *Load balancing* membagi jumlah pekerjaan yang harus dilakukan pada sebuah komputer menjadi dua atau lebih sehingga banyak pekerjaan dapat dilakukan secara bersamaan dan secara umum semua pengguna dapat dilayani lebih cepat. *Load balancing* dapat diimplementasikan dengan perangkat keras, perangkat lunak, atau kombinasi keduanya. *Load balancing* memberikan jaminan kualitas layanan (QoS) dalam *cloud computing*.

Dalam penelitian ini akan dilakukan implementasi *Load Balancing as a Service* (LBaaS) berbasis OpenStack. Kemudian juga akan dilakukan analisa perbandingan antara *single server* dan yang menggunakan *load balancing* dengan metode *round robin*, *least connection*, dan *source ip* dengan parameter pengukuran *load balancing* berupa *throughput*, *elapsed time*, *response time*, *transaction rate*, dan *cpu usage* dengan layanan *web server*.

2. Dasar Teori

2.1 Cloud Computing

Teknologi *Cloud Computing* merupakan suatu teknologi komputasi dimana semua *resource* dan sumber daya komputer baik itu memori, aplikasi, prosesor, *network*, sistem operasi yang digunakan dihadirkan secara virtual dengan pola akses *remote* sehingga kita bisa mengakses layanan tersebut kapanpun dan dimanapun selama kita terhubung dengan jaringan internet. Ketersediaan *on-demand* sesuai kebutuhan, mudah untuk dikontrol, dinamik dan skalabilitas yang hampir tanpa limit adalah beberapa atribut penting dari *cloud computing* [3].

2.2 OpenStack

OpenStack adalah sistem operasi *cloud* yang mengontrol beberapa sumber daya berupa komputasi, penyimpanan, dan jaringan dalam sebuah *data center*, semua dikelola melalui sebuah *dashboard* yang memberikan administrator kontrol melalui *web interface* [4].

2.3 Load Balancing

Load Balancing adalah proses pendistribusian beban terhadap sebuah servis yang ada pada sekumpulan *server* atau perangkat jaringan. Ketika ada permintaan dari pemakai ketika banyak permintaan dari pemakai maka *server* tersebut akan terbebani karena harus melakukan proses pelayanan terhadap permintaan pemakai. Solusi yang cukup bermanfaat adalah dengan membagi-bagi beban yang datang ke beberapa *server*, jadi tidak berpusat ke salah satu perangkat jaringan saja. Teknologi itulah yang disebut Teknologi *Load Balancing*. Teknologi *Load Balancing* dapat diperoleh keuntungan seperti menjamin reabilitas servis, availabilitas dan skalabilitas suatu jaringan [5].

2.4 Web Server

Web server atau biasa yang disebut dengan HTTP server karena menggunakan basis protokol *Hypertext Transfer Protocol* (HTTP). *Web server* merupakan *software* yang berfungsi untuk melayani permintaan HTTP request dari *web browser* dan menyediakan layanan akses ke suatu berkas, berkas tersebut dapat berupa *Hypertext Markup Language* (HTML), berkas *Javascript*, dan berkas *Perl* [6].

2.5 Nginx

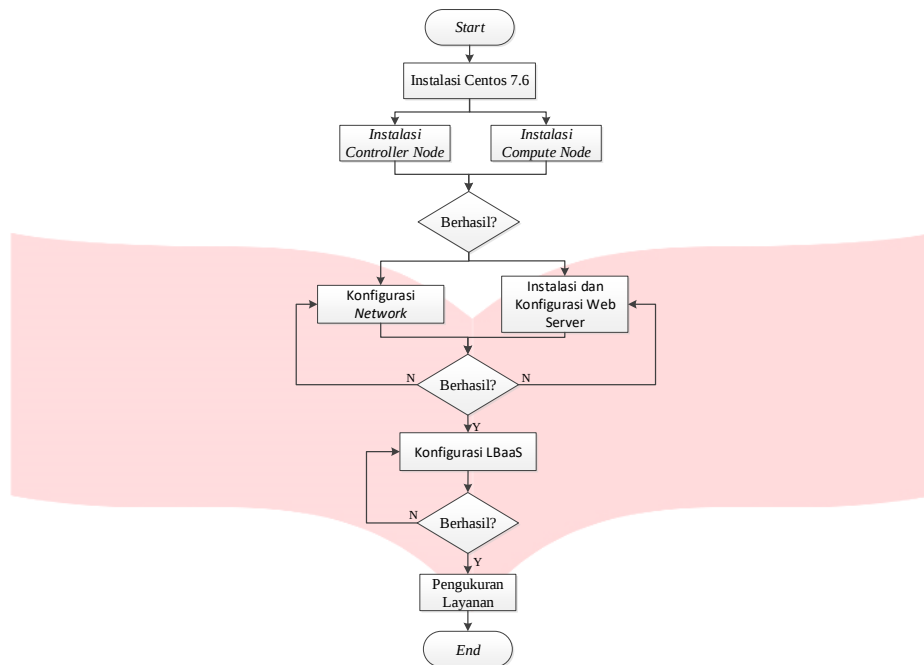
Nginx adalah *software open-source* yang memiliki kinerja tinggi sebagai *server HTTP* dan *reverse proxy*. Nginx dengan cepat memberikan konten statis dengan penggunaan efisien sumber daya sistem. Hal ini dapat menyebarkan dinamis HTTP konten di jaringan menggunakan *FastCGI handler* untuk *script*, dan dapat berfungsi sebagai perangkat lunak yang sangat mampu menyeimbangkan beban. Nginx dibangun secara modular dan dengan demikian mampu mendukung berbagai fitur seperti *Load Balancing* dan *Reverse Proxying*, *Virtual hosts* berbasis nama dan IP, *Fast CGI*, akses langsung ke cache, *SSL*, *Flash Video Streaming* dan sejumlah fitur-fitur standar lainnya. Nginx dapat dijalankan dan tersedia untuk platform Unix, Linux, varian dari BSD, MacOS X, Solaris, dan Microsoft Windows [7].

2.6 Siege

Siege adalah *tools benchmarking* yang digunakan untuk menguji http/https. Siege didesain agar seorang *web developers* dapat mengukur performansi *code* mereka dibawah tekanan, untuk melihat bagaimana *web* tersebut dapat bertahan di internet. Siege memungkinkan pengguna untuk melakukan permintaan request pada *web server* dengan jumlah *user* simultan yang dapat dikonfigurasi [8].

3. Perancangan Sistem

3.1 Diagram Alir Perancangan Sistem

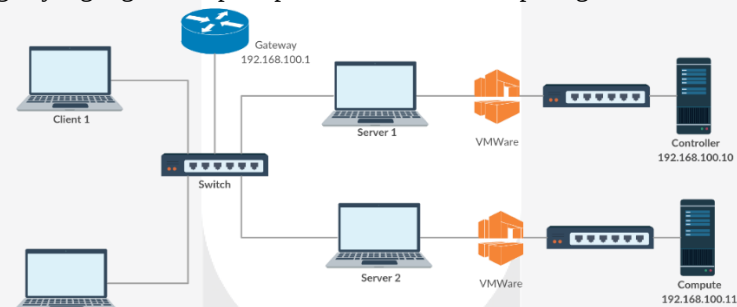


Gambar 3.1 Diagram Alir Perancangan Sistem

3.2 Desain Topologi Jaringan

Dalam topologi jaringan pada penelitian ini menggunakan 2 buah laptop yang masing-masing telah terinstall *virtual machine* berupa VMWare sebagai *server* OpenStack, 2 buah laptop untuk *client* yang diinstall aplikasi *siege* sebagai *load testing* untuk *web server* dan satu buah *switch* konvensional sebagai penghubung antar laptop.

Topologi jaringan yang digunakan pada penelitian ini adalah seperti gambar dibawah ini:



Gambar 3.2 Desain Topologi

Keterangan:

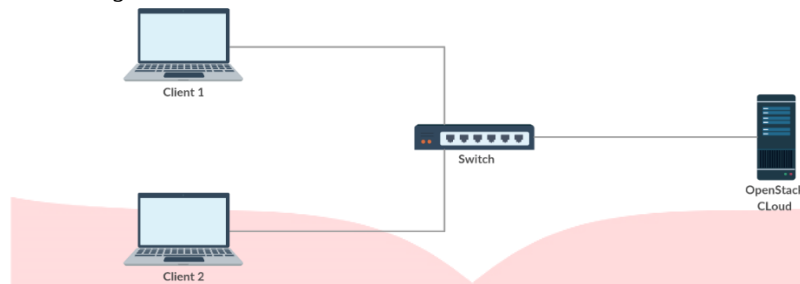
1. Server 1 dan Server 2 merupakan satu kesatuan dari *server* openstack yang terdiri dari dua *node*. Server 1 bertindak sebagai *controller* dan server 2 bertindak sebagai *compute*. Kedua server tersebut diinstall pada sistem operasi CentOS 7 yang berjalan diatas *virtual machine* berupa VMWare.
2. Client 1 dan Client 2 bertindak sebagai *load tester* yang telah terinstall aplikasi *siege* di dalam sistem operasi CentOS 7 yang berjalan diatas *virtual machine* berupa VMWare.
3. NIC virtual pada masing-masing VM dengan adapter bridge.

3.3 Skenario Pengujian

Skenario pengujian ini bertujuan untuk mengukur kinerja layanan *Load Balancer* pada OpenStack dengan layanan *web server* sehingga mendapatkan parameter pengukuran. Parameter-parameter pengukuran pada penelitian ini adalah untuk mendapatkan nilai seperti berikut:

1. *Throughput*,
2. *Elapsed Time*,
3. *Response Time*,
4. *Transaction Rate*,

5. CPU Usage,



Gambar 3.3 Topologi Pengujian

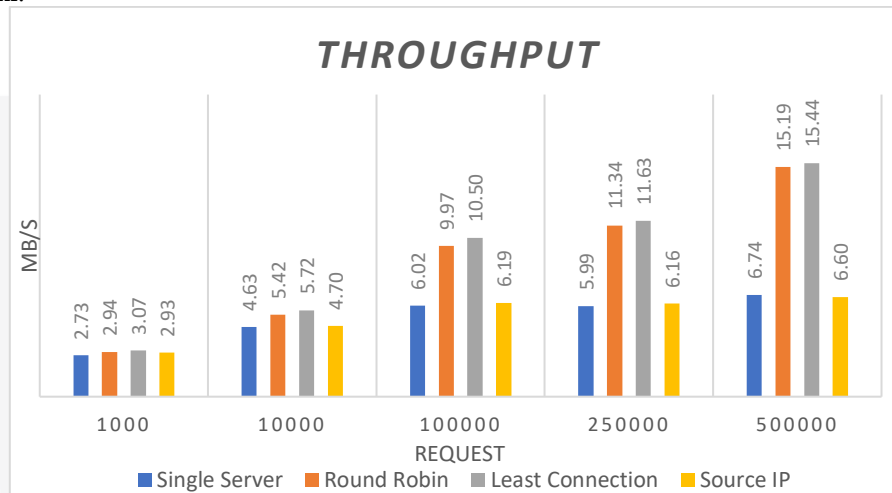
Untuk mendapatkan nilai seperti *throughput*, *elapsed time*, *response time*, *transaction rate*, dan *cpu usage* penulis melakukan percobaan berupa, client membangkitkan request HTTP sebesar 1000, 10.000, 100.000, 250.000, dan 500.000 dari aplikasi siege kepada *web server* yang kemudian hasil akan dapat dilihat setelah aplikasi siege berjalan dengan baik.

4. Hasil dan Analisis

Berikut ini adalah hasil dari 30 kali pengambilan data pada setiap parameter yang diuji:

4.1 Throughput

Throughput adalah jumlah rata-rata *byte* yang ditransfer setiap detik dari *server* ke semua pengguna yang disimulasikan [8]. Didapatkan nilai *throughput* dari pengujian pada *openstack cloud* pada grafik dibawah ini.

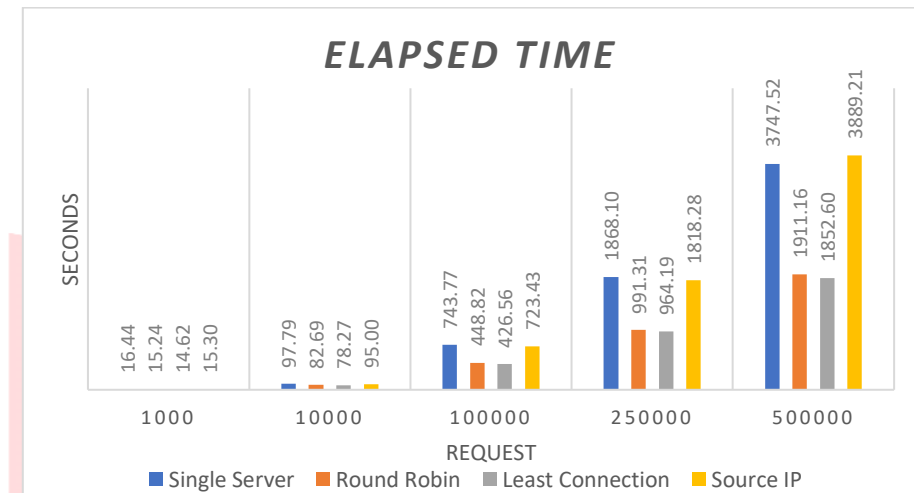


Gambar 4.1 Grafik *Throughput*

Dari grafik diatas menunjukkan bahwa nilai *throughput* cenderung naik seiring bertambahnya jumlah *request* yang diberikan. Namun pada saat server diberi beban 500.000 *request*, *single server* maupun *source ip* mengalami penurunan, dikarenakan server tersebut sudah tidak lagi mampu untuk menerima *request* sebanyak 500.000 tersebut, sehingga server sudah tidak dapat berjalan dengan optimal. Perbedaan yang signifikan antara *single server* dan *load balancing* terjadi ketika sistem diberi beban mulai dari 100.000 *request*. Hal tersebut terjadi karena sistem *single server* sudah tidak dapat menangani permintaan dengan baik sehingga *load balancing* lebih unggul.

4.2 Elapsed Time

Elapsed time adalah durasi keseluruhan tes yang dijalankan oleh siege dari *client* [8]. Didapatkan nilai *elapsed time* dari pengujian pada *openstack cloud* seperti grafik dibawah ini.

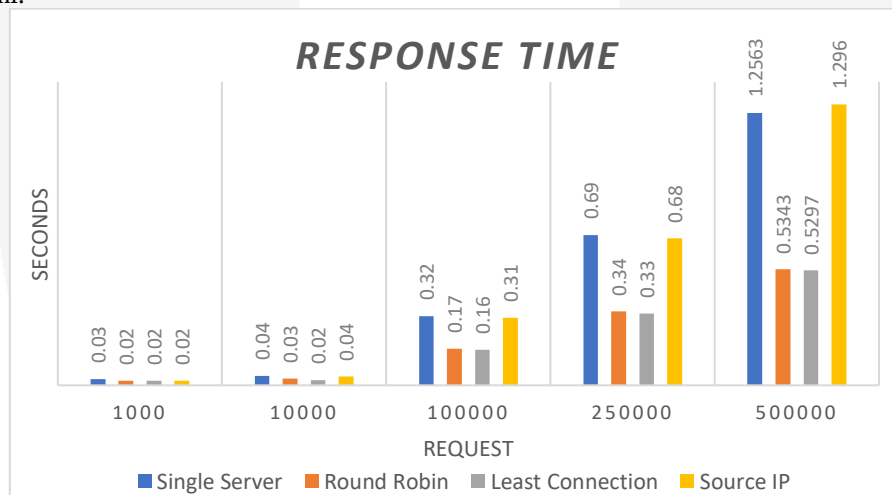


Gambar 4.2 Grafik *Elapsed Time*

Semakin rendah nilai *elapsed time*, maka performa sistem tersebut semakin bagus. Hasil pengujian ini juga sama seperti pengujian *throughput*, dimana hasil dari sistem *single server* dan sistem algoritma *source ip* cenderung sama. Kemudian untuk perbandingan hasil pengujian dari sistem *single server* dan sistem *load balancing* dengan algoritma *round robin* dan *least connection* terjadi perbedaan yang signifikan ketika sistem diberi beban mulai dari 100.000 *request* seperti yang dijelaskan pada sub-bab 4.1.1. Perbedaan yang signifikan ini dikarenakan baik sistem *single server* maupun sistem algoritma *source ip* hanya melayani *request* hanya dengan menggunakan satu *server*. Sedangkan sistem yang menggunakan algoritma *round robin* dan *least connection* membagi beban tersebut kepada ketiga *server* yang tersedia.

4.3 Response Time

Reponse time adalah waktu rata-rata yang diperlukan untuk menanggapi *request* dari setiap pengguna yang disimulasikan [8]. Didapatkan nilai *response time* dari pengujian pada *openstack cloud* seperti grafik dibawah ini.

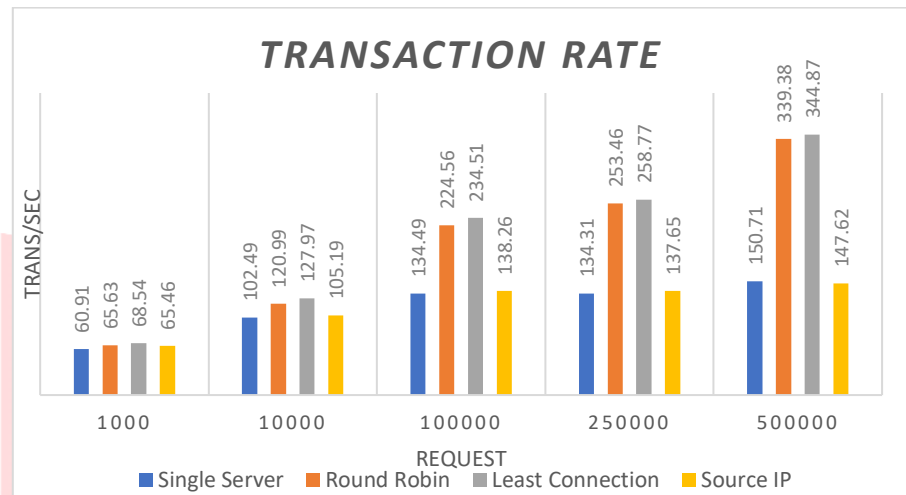


Gambar 4.3 Grafik *Response Time*

Pada pengujian ini semakin kecil nilai *response time* yang didapat maka semakin bagus performa sistem tersebut. Dari grafik tersebut dapat kita lihat bahwa sistem *single server* dan sistem algoritma *source ip* memiliki nilai *response time* yang lebih besar jika dibandingkan dengan sistem *load balancing* dengan algoritma *round robin* dan *least connection*, hal tersebut dikarenakan pada sistem *single server* dan *source ip* hanya menggunakan satu buah *web server* oleh sebab itu waktu yang dibutuhkan lebih lama untuk menyelesaikan *request* dari *client*.

4.4 Transaction Rate

Transaction rate adalah jumlah rata-rata transaksi yang dapat ditangani *server* per detik [8]. Berikut adalah grafik hasil dari pengujian parameter *transaction rate* pada *openstack cloud*.

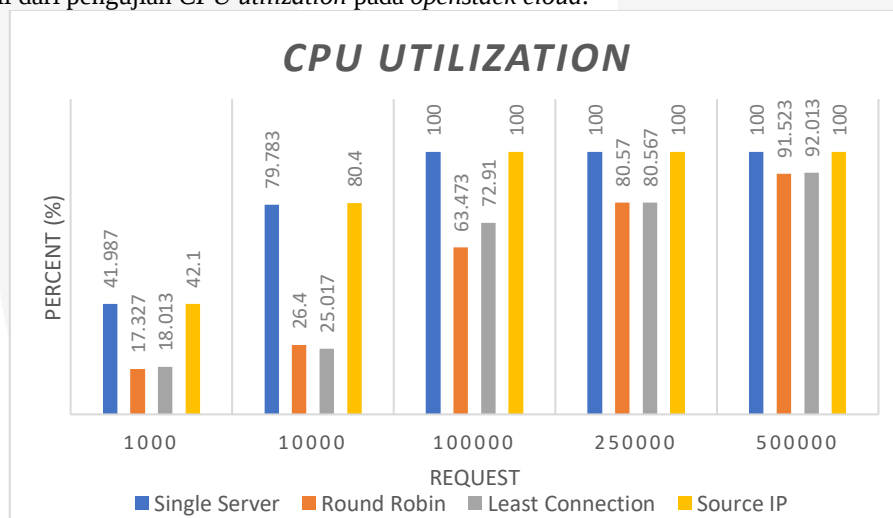


Gambar 4.4 Grafik *Transaction Rate*

Dari grafik diatas menunjukkan bahwa nilai *throughput* cenderung naik seiring bertambahnya jumlah *request* yang diberikan. Namun pada saat server diberi beban 500.000 *request*, *single server* maupun *source ip* mengalami penurunan, dikarenakan server tersebut sudah tidak lagi mampu untuk menerima *request* sebanyak 500.000 tersebut, sehingga server sudah tidak dapat berjalan dengan optimal. Dapat kita lihat bahwa jumlah *request* yang dapat dilayani per satuan detik pada sistem yang menggunakan *load balancing* dengan algoritma *round robin* dan *least connection* jauh lebih besar dibandingkan dengan sistem *single server* dan sistem algoritma *source ip*.

4.5 CPU Usage

Tujuan dari pengujian ini adalah untuk mengukur seberapa besar penggunaan *resource* dari segi CPU (*processor*) pada saat menjalankan sistem dengan *load balancing* dan *single server*. Berikut ini adalah grafik hasil dari pengujian *CPU utilization* pada *openstack cloud*.



Gambar 4.5 Grafik *CPU Utilization*

Jika persentase penggunaan CPU yang dihasilkan semakin kecil, maka sistem tersebut semakin bagus. Pada grafik tersebut dapat kita lihat bahwa sistem yang menggunakan *load balancing* lebih rendah dibandingkan dengan sistem *single server* dan *source ip*. Hal tersebut dikarenakan setiap *server* memiliki 1 *vcpu*, berbeda dengan sistem *load balancing* yang menggunakan algoritma *round robin* dan *least connection*, beban dari *client* dibagi kepada tiga *server* yang berarti tiga *vcpu* sehingga kinerja dari masing-masing *vcpu* menjadi lebih ringan dan hasil pengujian pun juga menjadi lebih optimal.

4.6 Fairness Index

Pengukuran *fairness index* bertujuan untuk mengetahui apakah setiap *web server* yang dijalankan pada *openstack* telah menerima sumber daya yang adil dan merata. Berikut adalah data statistik pembagian beban dari pengujian *load balancing* algoritma *round robin* dan *least connection* layanan web pada *openstack cloud*.

Tabel 4.1 Data Statistik Beban *Round Robin*

Server	Jumlah Request				
	1000	10000	100000	250000	500000
1	333	3333	33334	83333	166667
2	334	3333	33333	83334	166666
3	333	3334	33333	83333	166667

Tabel 4.2 Data Statistik Beban *Least Connection*

Server	Jumlah Request				
	1000	10000	100000	250000	500000
1	336	3351	32381	82809	168430
2	331	3336	33973	83302	166191
3	333	3313	33646	83889	165379

Dari Tabel 4.1 dan Tabel 4.2 diatas dapat kita lihat bahwa pembagian beban pada algoritma *load balancing round robin* lebih *fair* daripada pembagian beban pada algoritma *load balancing least connection*.

5. Kesimpulan dan Saran

5.1 Kesimpulan

Berdasarkan hasil dari penelitian yang telah dilakukan oleh penulis, maka dapat ditarik kesimpulan sebagai berikut:

1. *High-Availability* pada *web server* dapat diwujudkan dengan menerapkan *load balancing* dengan algoritma *round robin* dan *least connection*. Karena jika dilihat dari segi performansi, sistem yang menggunakan *load balancing* lebih unggul daripada sistem *single server* maupun *source ip*.
2. *Source IP* membagi beban berdasarkan ip sumber atau ip *client*. Oleh sebab itu hasil dari pengukuran *single server* dan *source ip* cenderung sama dikarenakan ip sumber hanya berasal dari satu *client*.
3. Perbedaan yang signifikan antara *single server* dan *load balancing* terjadi pada saat sistem diberi beban mulai dari 100.000 *request*. Hal ini terjadi karena pada saat sistem diberi beban 1000 dan 10.000 *request*, sistem *single server* masih dapat melayani permintaan dengan baik.
4. Pada pengujian fairness, algoritma *round robin* lebih *fair* dalam pendistribusian beban dibanding dengan algoritma *least connection* yaitu dengan nilai di masing-masing server yang hampir sama. Sedangkan pada algoritma *least connection* mempunyai selisih yang cukup banyak di masing-masing server.
5. Pada pengujian layanan web, sistem *load balancing* menunjukkan hasil berupa *throughput*, *elapsed time*, *response time*, *transaction rate*, dan *cpu usage* lebih unggul jika dibandingkan dengan sistem *single server*. Hal ini ditunjukkan dengan peningkatan nilai *throughput* sebesar 65,61% dari *single server*, penurunan nilai *elapsed time* sebesar 46.93% dari *single server*, penurunan nilai *response time* sebesar 46.87% dari *single server*, peningkatan nilai *transaction rate* sebesar 66,97% dari *single server*, dan penurunan nilai *cpu utilization* sebesar 58.74% dari *single server*.

5.2 Saran

Saran yang dapat diusulkan pada penelitian ini adalah:

1. Menggunakan lebih dari 3 *instance* sebagai perbandingan jumlah *web server*.
2. Menggunakan *operator-scale load balancing opensource*, seperti Ocatavia.
3. Membandingkan dengan *load balancer* lain.

Daftar Pustaka

- [1] M. Yang, H. Wang, and J. Zhao, "Research on Load Balancing Algorithm based on the Unused Rate of the CPU and Memory", International Conference on Instrumentation & Measurement, Computer, Communication and Control, 2015.
- [2] V. M. Sivagami, K. S. Easwarakumar, "Performance analysis of Load balancing algorithms using LBaaS", International Journal of Research and Analytical Reviews, vol. 5, September 2018.
- [3] P. Mell, T. Grance, "The NIST Definition of Cloud Computing Recommendations of the National Institute of Standards and Technology", Nisc Spec. Publ, vol. 145, 2011.
- [4] OpenStack, [online] <https://www.openstack.org/software/> , Diakses pada tanggal 18 Februari 2019.
- [5] M. Arman, N. Wijaya, H. Irsyad, "Analisis Kinerja Web Server Menggunakan Algoritma Round Robin dan Least Connection", Jurnal SISFOKOM, vol. 06, no. 01, Maret 2017.
- [6] B. A. Forouzan, Data Communications and Networking, Fourth Edition, New York: McGraw-Hill, 2007.
- [7] A. Aziz, T. Tampati, "Analisis Web Server untuk Pengembangan Hosting Server Institusi: Perbandingan Kinerja Web Server Apache dengan Nginx", Jurnal Multinetics, vol. 1, no. 2, November 2015.
- [8] J. Fulmer, et al, "Designed and implemented Siege in his position as Webmaster for Armstrong World Industries", [online] <https://www.joedog.org/siege-manual/>, Diakses pada tanggal 20 September 2019.

