Vol.6 No.1 Tahun 2020
Jurnal Teknologi dan Manajemen Informatika
http:// http://jurnal.unmer.ac.id/index.php/jtmi
P-ISSN: 1693-6604 E-ISSN: 2580-8044

Implementasi *Load Balancing* Pada Web Server Menggunakan Nginx

Fahmi Apriliansyah [1], Iskandar Fitri [2], Agus Iskandar [3]

Fakultas Teknologi Komunikasi dan Informatika, Universitas Nasional

Info Artikel

Riwayat Artikel
Diterima: 20-12-2019
Disetujui: 19-03-2020

Kata Kunci
Apache;
Load balancing;
Nginx;
Overload;
Web Server

✉ Corresponding Author
Fahmi Apriliansyah
Fakultas Teknologi Komunikasi dan Informatika, Universitas Nasional
Tel. +62 8569091075
fahmi.aprilliansyah41@gmail.com

ABSTRAK

Kebutuhan internet mempengaruhi pengunjung website semakin meningkat dan membuat beban traffic meningkat pada server, semakin banyak jumlah *traffic* ke server dapat menyebabkan web server menjadi *down*, jika digunakan dengan jumlah berlebih (*overload*). Server *down* karena tidak mampu dalam menjalankan *request* yang berlebihan, upaya untuk meningkatkan kinerja web server karena adanya akses *request* layanan yang sangat banyak, maka menggunakan sistem *load balancing* adalah solusi untuk mengatasi terjadinya server *down*. Server *load balancing* dapat bertugas untuk mendistribusikan beban kerja ke banyak server, dengan mempertimbangkan kapasitas dari setiap server yang ada. Pengujian ini dilakukan dengan 6 buah server. Yang masing masing mendapatkan 4000, 8000, 12000, 16000 *request*, dengan pengujian menggunakan *software* apache *benchmarking tool*. Dengan menerapkan *load balancing* Nginx mampu menstabilkan dan menjaga keseimbangan web server, dengan didukung 3 metode algoritma (Round Robin, Leas Connection, IP Hash) yang dapat digunakan dalam *load balancing* Nginx. Metode algoritma yang terbaik adalah Least Connection karena mempunyai stabilitas *response time* 116ms yang signifikan dan kecepatan *response time* lebih bagus dan mendapatkan 2300.96 req/s.

PENDAHULUAN

Hasil studi pada tahun 2018 yang dilakukan oleh Polling Indonesia bekerja sama dengan Asosiasi Penyelenggara Jasa Internet Indonesia (APJII) didapatkan total jumlah pengguna internet di Indonesia telah mencapai 171,17 juta pengguna, yang mana angka tersebut setara dengan total 64,8% penduduk Indonesia. Pesatnya perkembangan internet mempengaruhi beban *traffic* sebab server digunakan dengan jumlah yang berlebih (*overload*). Server *overload* akan mengakibatkan sever *down* karena tidak mampu dalam menjalankan *request*, maka dari itu penggunaan sistem *load balancing* diperlukan untuk mengatasi terjadinya server *down*, sistem *load balancing* dapat bertugas untuk mendistribusikan beban kerja ke banyak server, dengan mempertimbangkan kapasitas dari setiap server yang ada (Dani & Suryawan, 2012).

Sistem *load balancing* dapat bekerja dengan baik ketika *request* datang dari klien telah berhasil didistribusikan *balancer* secara merata kepada setiap *node cluster*. Sehingga server tidak mengalami *overload* dan kemampuan web server bisa melayani 10.000 *request* dengan tidak mengalami *error request* (Rahmatulloh & MSN, 2017). Penggunaan teknik *load*

balancing menjadi suatu pilihan solusi teknologi yang sangat efektif untuk memanfaatkan *bandwith* internet tanpa harus ketimpangan (Warman & Andrian, 2017). Sedangkan kemampuan sistem *network load balancing* dalam melayani *request* lebih besar dibandingkan dengan *single* server. Sistem *network load balancing* mampu melayani *request* maksimal 10,82 per detik sedangkan pada *single* server hanya mampu melayani maksimal 6,25 *request* per detik sehingga pada sistem *network load balancing* memiliki *throughput* yang lebih bagus dibandingkan *single* server. Respon sistem *network load balancing* lebih cepat saat jumlah request dibangkitkan 1000 adalah 333716,915 mili detik sedangkan waktu respon untuk sistem *single* server yaitu 58809,28 mili detik sehingga antrian *request* yang terjadi pada sistem *network load balancing* tidak mengalami antrian panjang dibandingkan *single* server (Yusuf, Riza, Ariefianto, & Elektro, 2013).

Berdasarkan penjelasan diatas, Nginx merupakan web server yang ringan dan memiliki perfoma yang cepat, Nginx dapat memproses beberapa *reques*t dengan baik. Nginx dapat diakses oleh banyak klien diwaktu yang bersamaan yang artinya server Nginx mampu menerima banyak *traffic* dengan menggunakan *load balancing Nginx*. Hasil pada penelitian ini didapatkan dengan algoritma yang diuji yaitu Round Robin, *Least Connection* dan Ip Hash dengan perbandingan dari parameter *throughput*, *response time*, *request per second*, *CPU Utilization* dan *downtime*. Berdasarkan hasil penelitian didapatkan pengujian dengan menggunakan *Least Connection* merupakan algoritma terbaik berdasarkan paramater yang telah diujikan.

METODE

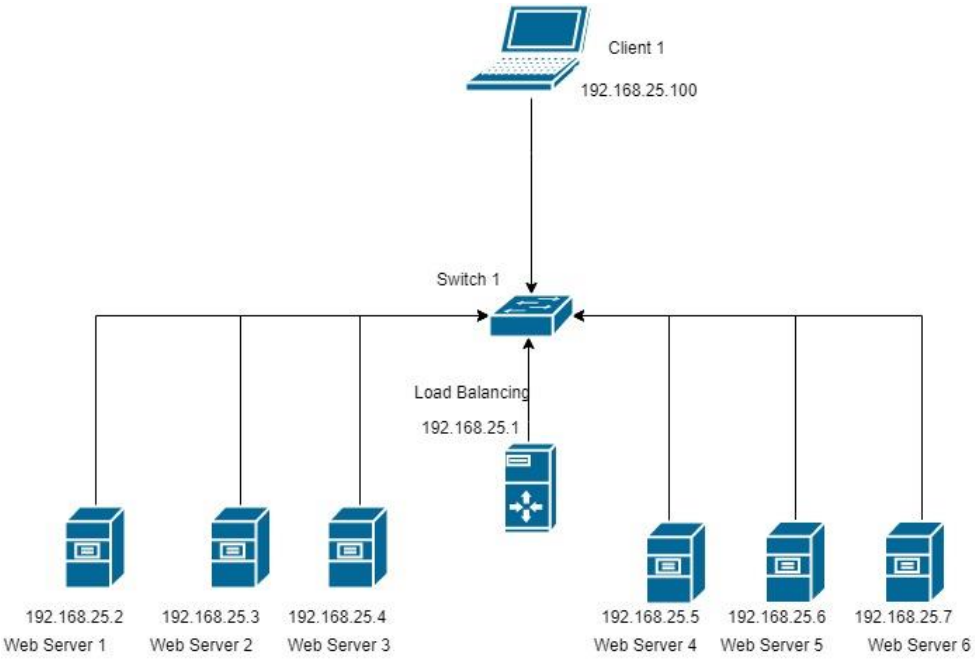
Rancangan sistem *load balancing* dalam penelitian ini menggunakan 7 buah komputer yang berperan sebagai klien dan server. Dari 7 komputer tersebut, 1 di antaranya berperan sebagai *load balancing*, 6 sebagai server (Riskiono, 2018). Ada 3 metode *load balancing* yang didukung oleh Nginx.

1. *Round Robin*
   Round Robin merupakan algoritma paling sederhana dan paling banyak digunakan oleh perangkat load balancing. *Round Robin* bekerja dengan cara membagi beban secara bergiliran dan berurutan dari satu server ke server lainya (Diarjo & Mulyana, 2017).
2. *Least Connection*
   Least Connection melakukan pembagian beban berdasarkan banyaknya koneksi yang sedang dilayani oleh sebuah server. Server dengan koneksi yang paling sedikit akan diberikan beban berikutnya (Arman, Wijaya, & Irsyad, 2017).
3. IP Hash
   Tidak seperti *Round Robin* dan *Least Connection* setiap request kemungkinan akan diarahkan pada server yang berbeda, pada IP Hash setiap *request* dari klien dipastikan diarahkan ke server yang sama berdasarkan alamat IP (Chandra, 2019).

Spesifikasi Perangkat

Terdapat 7 komputer yang digunakan dalam penelitian ini, 6 sebagai server, 1 sebagi *load balancing*. Ketujuh komputer tersebut yang digunakan sebagai server dan *load balancing* tersebut memiliki i7-8750 2.20 Ghz, 16GB DDR4. 1TB HDD dengan sistem operasi Centos 8, dengan metode virtual server. Sedangkan komputer klien memiliki spesifikasi hardware sebagai berikut Intel Core i7-8750 2.20Ghz, 32GB, 1TB HDD dengan sistem operasi Windows 10. Untuk software yang digunakan antara lain Nginx, *apache benchmark tools*, *bind* (Riskiono, 2018).

Perancangan Desain Sistem



Gambar 1. Desain Sistem

Pada Gambar 1 diatas merupakan rancangan untuk menggambarkan susunan sistem yang dipakai. Dalam desain ini *switch* dilambangkan sebagai penghubung jaringan, lalu pada uji coba terdapat 6 web server yang dihubungkan kedalam satu server *load balancing* yang didalamnya terdapat dns server (Pratama, Hafidudin, & Aulia, 2015).

Parameter Pengujian

Tahapan ini menjelaskan hasil dan pembahasan bagaimana jika server dilakukan akses dengan banyak koneksi, pada tahapan ini dilakukan pengujian menggunakan 1 client, 6 server dan aplikasi *apache- HTTP benchmarking tool*. Dengan hasil output *Throughput*, *Response time*, *Request Second* dan CPU *Utilization* (Rosalia, Munadi, & Mayasari, 2016).

1. *Throughput* adalah bandwiwdth actual yang terukur pada ukuran waktu tertentu di jaringan. Pengujian *throughput* merupakan parameter variable dari QoS yang ditunjukan untuk melihat performa jaringan pada kecepatan pengiriman paket yang dilakukan dengan mengirim dan memanfaatkan bandwidth yang ada (Darma & Atitama, 2017).
2. *Response time* ditunjukan guna mengukur seberapa secpat suatu response dapat menerima request dari klien (Balance, Performace, Apache, Server, & Database, n.d.).
3. *Request per second* dilakukan dengan mengirim layanan per satuan detik dari client yang kemudian dikirmkan ke web server untuk mengetahui kinerja server yang menggunakan sistem *load balancing* (Adnan, 2016).
4. *Downtime* digunakan untuk merujuk pada periode ketika sistem tidak tersedia. Durasi *downtime* atau outage mengacu pada periode waktu dimana sistem gagal menjalankan fungsi server utama (Sirajuddin, Affandi, & Setijadi, 2012).
5. CPU *Utilization* pengujian dilakukan untuk meninjau penggunaan resources oleh centos 8 dari sisi CPU (Processor) untuk melihat perbedaan penggunakan CPU pada setiap centos 8 saat menjalankan *load balancing* (Chen, Noertjahyana, & Andjarwirawan, n.d.).

Tabel 1 Spesifikasi Server

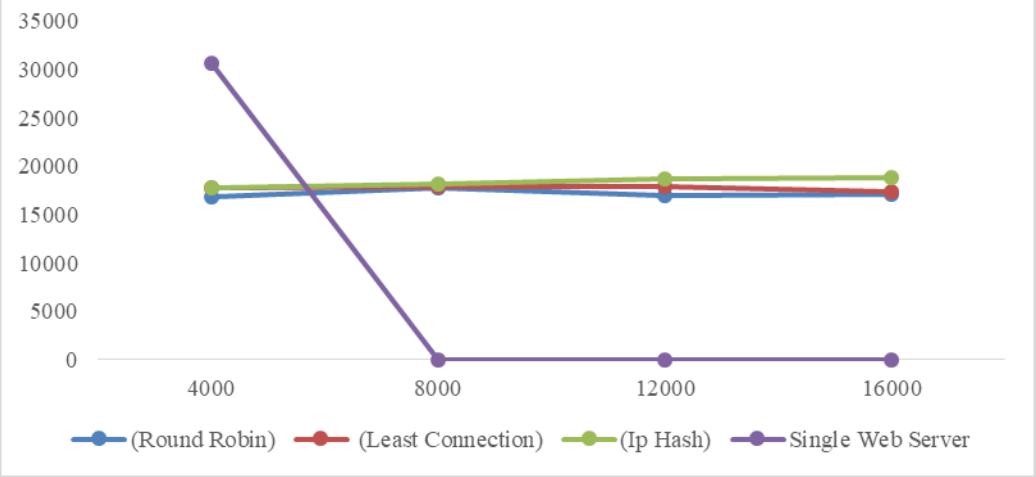
Host Name	IP Address	Operating System	Memory	CPU
Load Balancing	192.168.25.1	Centos 8	8 GB	1 CORE
Server 1	192.168.25.2	Centos 8	1 GB	1 CORE
Server 2	192.168.25.3	Centos 8	1 GB	1 CORE
Server 3	192.168.25.4	Centos 8	1 GB	1 CORE
Server 4	192.168.25.5	Centos 8	1 GB	1 CORE
Server 5	192.168.25.6	Centos 8	1 GB	1 CORE
Server 6	192.168.25.7	Centos 8	1 GB	1 CORE

Tabel 1 diatas merupakan spesifiikasi Yang diperlukan dalam membuat sistem yang diperlukan dengan sistem operasi centos 8 dan mengetahui berapa server yang digunakan dalam penelitian ini. Semua server akan diisi dengan web server Nginx dengan memasukan ip berbeda. Dengan menjalanan serangkaian test dan uji coba dengan aplikasi Nginx di *virtual machine* dengan metode *load balancing* yang akan melakukan traffic management pada web server.

HASIL DAN PEMBAHASAN

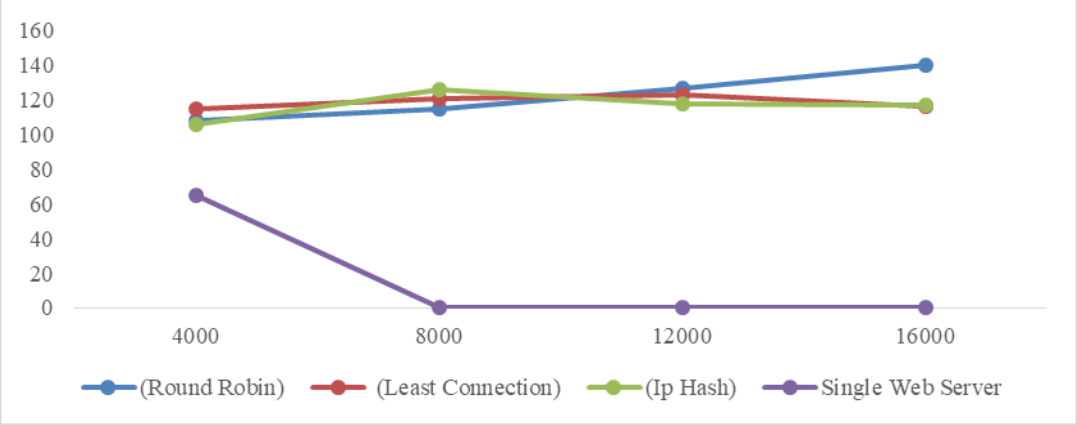
Hasil Pengujian Melalui Apache HTPP Benchmarking Tools

Pada tahapan ini dilakukan pengetesan *Transfer rate* waktu akses yang dilakukan secara bersamaan sebanyak 4000, 8000, 12000, 16000 *request* dan 200 klien. Hasil yang akan didapatkan di sini adalah *Throughput, Response time,* Req Sec, dan CPU *Utilization*.



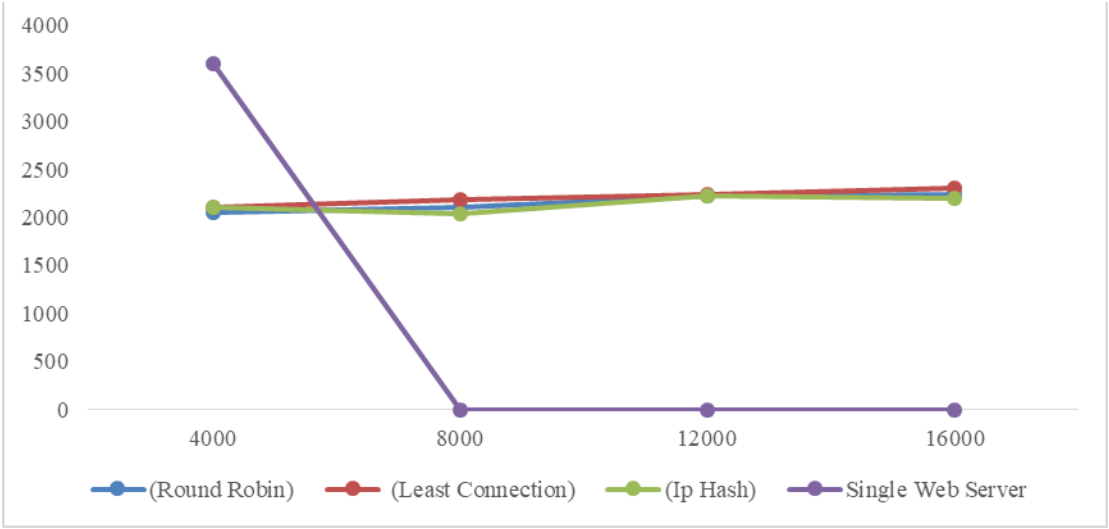
Gambar 2. *Least Connection*

Pada gambar 2 dijelaskan bahwa pada grafik metode *Least Connection* terjadi stabilitas *throughput* pada kecepatan 17mbps, pada *Round Robin* terjadi varietas pada *throughput* yaitu antara 16mbps dan 17mbps. Pada metode IP Hash terjadi peningkatan yang signifikan pada *throughput* dimulai dari 17mbps pada 4000 *request* hingga 18.8mpbs pada 16000 *request*. Sedangkan pada single server terjadi kegagalan jika *request* yang di tampung semakin banyak, pada grafik tersebut diketahui bahwa metode ip hash lebih unggul pada kecepatan pengambilan data. Hal tersebut dikarenakan bahwa server yang diakses hanya satu dan jika server tersebut mati barulah server akan di ganti dengan server selanjutnya. Akan tetapi akan terlihat bahwa beban yang di tanggung server akan lebih banyak karena server hanya di akses satu per satu, lalu pada tempat ke dua yaitu metode *Least Connection.* Oleh sebab itu stabilitas yang terjadi stabil tanpa mengurangi koneksi yang terjadi, pada tempat ke tiga metode *Round Robin* yang mengandalkan pembagian server secara merata (bergantian) terjadi adanya peningkatan dan penurunan jika *request* lebih banyak karena banyaknya antrian *request* yang terjadi .



Gambar 3. *Response time*

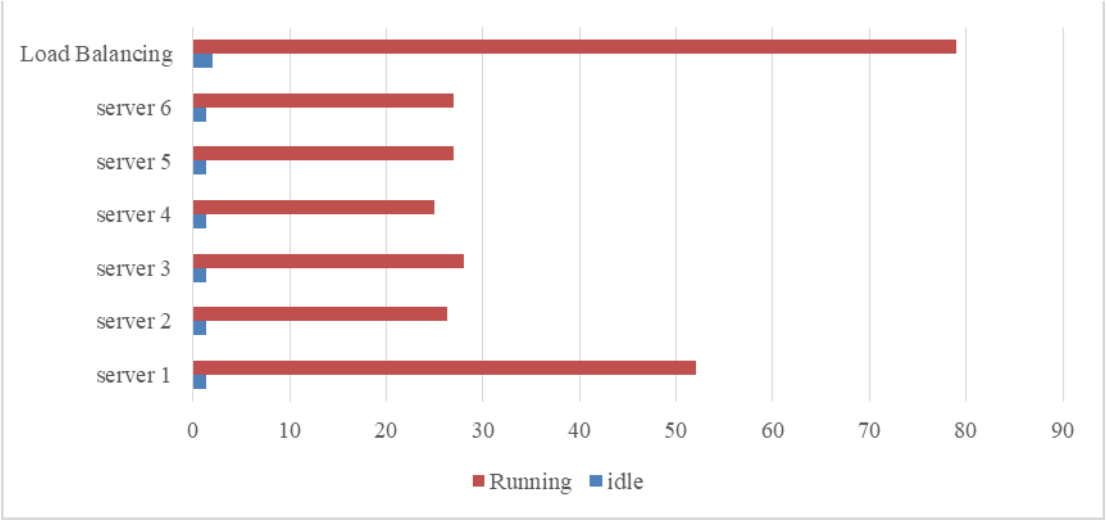
Pada gambar 3 menunjukan pergerakan yang terjadi pada masing masing server dengan pengetesan jumlah request 4000, 8000, 12000, 16000 *request* dan masing masing mendapatkan 200 client. Dari hasil *request* gambar 3 dapat dilihat pada metode *Round Robin* waktu *response time* akan lebih tinggi jika request yang diterima lebih banyak sebesar 140ms karena pada metode *Round Robin* mengandalkan perataan request yang di dapat pada setiap server maka antrian yang akan terjadi juga akan semakin besar jika request yang diterima semakin banyak. *Response time* pada metode *Least Connection* menunjukan bahwa metode ini menunjukan bahwa metode ini mempunyai stabilitas *response time* yang signifikan dan kecepatan *response time*nya lebih rendah sebesar 116ms dari pada *Round Robin*. Pada metode IP Hash *response time* lebih rendah di sedikit *request* karena server terbebani satu per satu dan jika server tidak dapat diakses barulah berpindah ke server lainnya IP Hash mendaptkan *response time* maksimal 117ms, pada single web server tidak dapat mengakses jika servernya terbebani terlalu banyak karena port yang di akses sudah habis.



Gambar 4. *Request persecond*

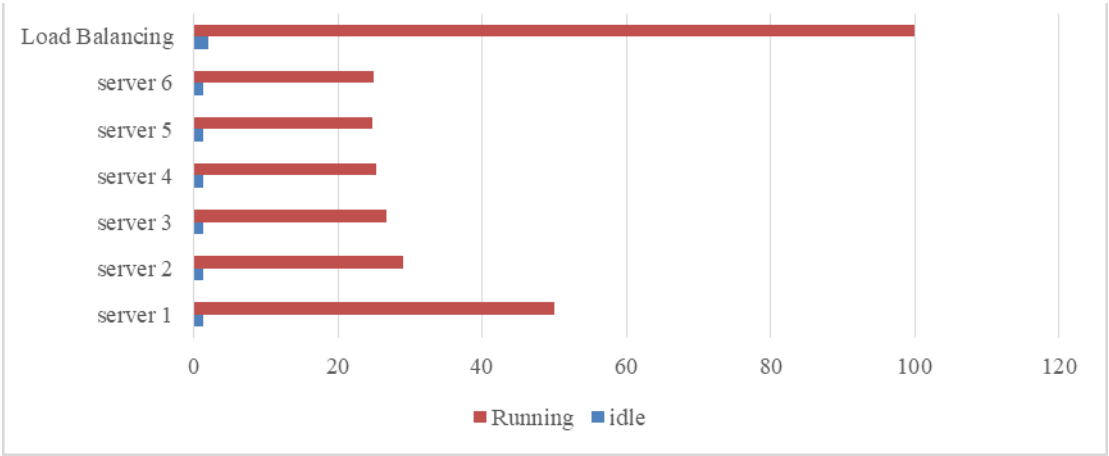
Pada gambar 4 terlihat bahwa jumlah request yang dapat dilayani server akan bertambah jika server dibebani oleh banyak *request*. Terlihat juga pada grafik bahwa pada metode *Least Connection* mendapatkan 2300.96 Req/s, sehingga lebih unggul pada sektor *request* perdetik yang dapat ditanggung server pada metode lain yaitu *Round Robin* mendapatkan 2235.36 Req/s dan terjadi penurunan performa pada permintaan *request* yang lebih besar karena pada metode *Round Robin* mengantrikan servernya sehingga *request* harus menunggu. Sedangkan metode lainnya Ip Hash mendapatkan 2202.26 Req/s lebih kepada pembebanan satu server sehingga server yang di load akan menujukan penurunan pada request yang lebih tinggi dengan client yang lebih tinggi karena *request* tidak secara langsung dibagikan ke masing -

masing server, dan kemungkinan server akan *not responding* lebih tinggi jika hanya menggunakan sedikit web server.



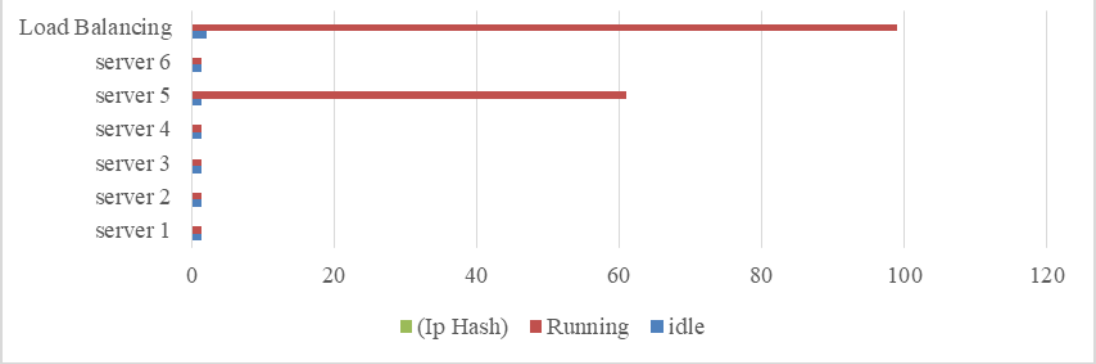
Gambar 5. CPU *utilization* pada *Round Robin*

Pada Gambar 5 tertera CPU *Utiilization* digunakan service Nginx pada saat melakukan pengujian request dengan 200 *client*. Secara bersamaan pada keenam *web server* dengan metode *Round Robin*. Server *load balancing* tersebut, pada gambar 5 menunjukan bahwa CPU *Utilization* untuk mengakses Nginx dengan metode *Round Robin* pada server 1 digunakan 52% CPU *Utilization* karena server 1 merupakan server pertama yang diakses oleh *load balancing* server 2 menunjukan 26.3% *usage* begitu juga dengan server 3, 4, 5, 6 rata - rata menunjukan tidak lebih dari 28% CPU *Utilization*, sedangkan CPU *Utilization load balancing* 79% digunakan.



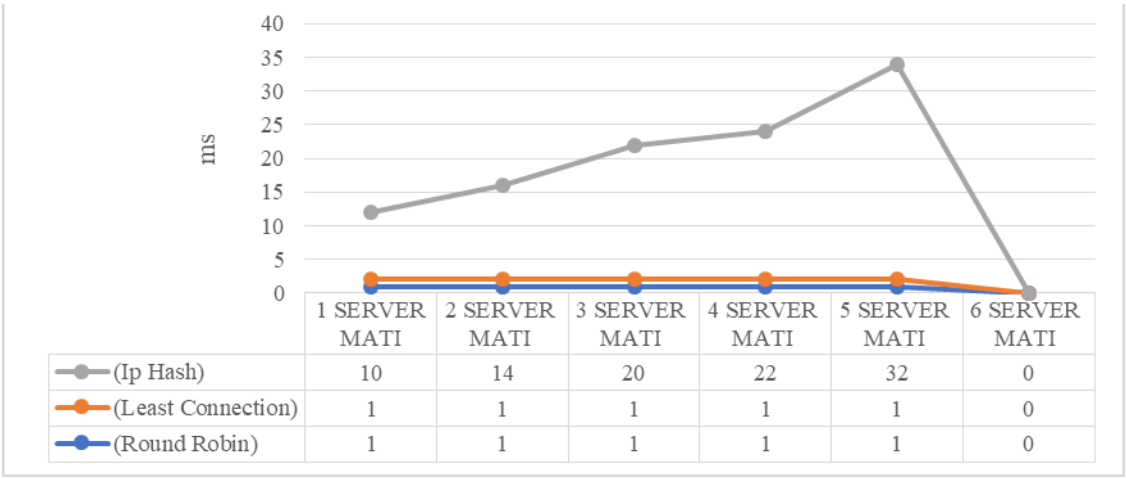
Gambar 6. CPU *utilization* pada *Least Connection*

Pada gambar 6 tertera CPU *Utilization* di gunakan service Nginx pada saat melakukan pengujian *request* dengan 200 klien secara bersamaan pada keenam *web server* dengan metode *Least Connection*. Server *load balancing* dan pada gambar 6 menunjukan bahwa CPU *Utilization* untuk mengakses Nginx dengan metode *Least Connection* pada server 1 digunakan 50% server usage karena server 1 merupakan server pertama yang diakses oleh *load balancing.* Server 2 menunjukan 29% akan tetapi *usage* pada server 3, 4, 5, 6 rata-ratanya turun dan menunjukan tidak lebih dari 26% CPU *Utilization*. Tetapi CPU *Utilization load balancing* 100% digunakan, karena di metode ini *load balancing* yang menentukan server mana yang sedang tidak mendapat *request*.



Gambar 7. CPU *utilization* pada IP Hash

Pada gambar 7 tertera CPU *Utilization* di gunakan *service* Nginx pada saat melakukan pengujian *request* dengan 200 client secara bersamaan pada keenam web server dengan metode IP Hash. Server *load balancing*, Pada gambar 7 menunjukan bahwa Cpu *Utilization* untuk mengakses Nginx dengan metode IP Hash pada server 1 digunakan 1.3% sama seperti saat idle, server 2,3,4,6 juga menunjukan demikian yaitu 1.3% CPU *Utilization* akan tetapi *usage* pada server 5 CPU *Utilization* yang digunakan 61% CPU *Utilization*, karena pada IP Hash server yang di gunakan klien akan tetapi sama sampai server tersebut tidak dapat dijangkau, lalu CPU *Utilization* pada *load balancing* mencapai hingga 99% CPU *Utilization*, karena *load balancing* terus berkomunikasi dengan web server yang akan menampilkan untuk klien.



	1 SERVER MATI	2 SERVER MATI	3 SERVER MATI	4 SERVER MATI	5 SERVER MATI	6 SERVER MATI
(Ip Hash)	10	14	20	22	32	0
(Least Connection)	1	1	1	1	1	0
(Round Robin)	1	1	1	1	1	0

Gambar 8. *Downtime*

Pada Gambar 8 Menjelaskan tentang terjadinya *downtime* server pada *Round Robin*, *Least Connection*, Ip Hash. Dengan uji coba server 1 sampai 6 server mati, dan *load balancing restart*. Pada *Round Robin* dan *Least Connection* mendapatkan 1ms karena *Round Robin* tidak terlihat *downtime* karena server akan terus memutari antrian server yang ada dan apabila ada server yang mati maka akan di loncati sehingga *response time* yang ada pun tidak lebih dari 20ms, sedangkan *Least Connection* tidak terlihat *downtime* karena server akan memilih server yang paling cepat diakses. *Downtime* paling tinggi pada Ip Hash mendapatkan 32ms karena tidak seperti *Round Robin* dan *Least Connection* setiap request kemungkinan akan diarahkan pada server yang berbeda, pada IP Hash setiap request dari klien dipastikan diarahkan ke server yang sama berdasarkan alamat IP. Jika salah satu server sudah di akses klien maka server tersebut akan terus di akses oleh klien tersebut sampai server tidak bisa di akses. Ketika *load balancing* di *restart* paling rendah pada *Round Robin* mendapatkan 480ms dan tertinggi IP Hash 944ms.

SIMPULAN DAN SARAN

Berdasarkan dari pengujian web server, kemampuan sistem *load balancing Nginx* dengan 3 metode algoritma *Round Robin*, *Least Connection*, dan IP Hash yang terbaik adalah *Least Connection*. Hasil yang didapatkan *Least Connection mendapatkan response time* 116ms, 2300.96 req/s *dan throughput 17380.01 kbps* Sedangkan *Round Robin* mendaptkan 140ms, 2335.36 req/s dan *throughput* 17098.05 kbps. Kemampuan sistem network *load balancing* Nginx dalam melayani *request* lebih besar dibandingkan dengan *single* server. Sebab sistem network *load balancing* Nginx mampu menjalankan 16000 request sedangkan single server hanya 4000. Dilihat dari segi kemampuan *load balancing* Nginx dengan enam buah web server mampu menstabilkan dan menjaga keseimbangan web server, yang mampu melayani traffic tinggi, di bandingkan dengan single web server. Berdasarkan pengujian yang dilakukan, pada 6 server *downtime* yang terjadi pada Nginx kurang dari 2ms karena Nginx telah mengalokasikan server yang ada sehingga klien di arahkan ke server lain yang aktif. Sedangkan jika mengunakan 1 server tanpa *load balancing*, jika server hang tidak akan ada server lainnya yang menanggulanginya

DAFTAR PUSTAKA

[1] Adnan, F. (2016). Analisis Perbandingan Performa Web Server Apache dan Nginx menggunakan Httperf pada VPS dengan Sistem Operasi CentOs. *Stmik Amikom Yogyakarta*, 6. https://doi.org/10.1103/PhysRevD.85.065021

[2] Arman, M., Wijaya, N., & Irsyad, H. (2017). Analisis Kinerja Web Server Menggunakan Algoritma Round Robin dan Least Connection. *Jurnal Sisfokom (Sistem Informasi Dan Komputer)*, *6*(1), 55. https://doi.org/10.32736/sisfokom.v6i1.143

[3] Balance, L., Performace, A., Apache, U., Server, N. W. E. B., & Database, W. P. (n.d.). *Performa Algoritma Load Balance Pada Server Web Apache Load Balance Algoritms Performace Using Apache and Nginx Web Server*. 1–6.

[4] Chandra, A. Y. (2019). *Analisis Performansi Antara Apache & Nginx Web Server dalam Menangani Client Request*. 48–56. https://doi.org/10.30864/jsi.v14i1.248

[5] Chen, W., Noertjahyana, A., & Andjarwirawan, J. (n.d.). *Analisis Perbandingan Kinerja Algoritma Load Balancer NGINX pada Studi Kasus PRS*.

[6] Dani, R., & Suryawan, F. (2012). Balancing Dan Failover Menggunakan Nginx. *Khazanah Informatika*, *3*(1), 43–50.

[7] Darma, I. M. A. S., & Atitama, I. G. O. G. (2017). Implementasi Load Balancing Pada Openstack dengan Metode Round Robin. *Prosiding Seminar Nasional Informatika (SENAPATI)*, (ISSN:2087-2658), 115–119.

[8] Diarjo, A. A., & Mulyana, D. I. (2017). Penerapan Algoritma Round Robin Dan Modulo Pada Load Balancing. *Jurnal CKI On SPOT*, *10*(1), 21–34.

[9] Pratama, M. R., Hafidudin, & Aulia, S. (2015). Analisis Performansi Load Balancing Dengan Algoritma Round Robin Dan Least Connection Pada Sebuah Web Server. *E-Proceeding of Applied Science*, *1*, *No.1*(ISSN:2442-5826), 1577–1585.

[10] Rahmatulloh, A., & MSN, F. (2017). Implementasi Load Balancing Web Server menggunakan Haproxy dan Sinkronisasi File pada Sistem Informasi Akademik Universitas Siliwangi. *Jurnal Nasional Teknologi Dan Sistem Informasi*, *3*(2), 241–248. https://doi.org/10.25077/teknosi.v3i2.2017.241-248

[11] Riskiono, S. D. (2018). Implementasi Metode Load Balancing Dalam Mendukung Sistem Kluster Server. *SEMNAS RISTEK*, 455–460.

[12] Rosalia, M., Munadi, R., & Mayasari, R. (2016). Implementasi High Availability Server Menggunakan Metode Load Balancing Dan Failover Pada Virtual Web Server Cluster. *E-Proceeding of Engineering*, *3*(3), 4496–4503.

[13] Sirajuddin, Affandi, A., & Setijadi, E. (2012). Rancang Bangun Server Learning Management System Menggunakan Load Balancer dan Reverse Proxy. *JURNAL TEKNIK ITS Vol., 1*, 50–52. https://doi.org/2301-9271

[14] Warman, I., & Andrian, A. (2017). Analisis Kinerja Load Balancing Dua Line Kineksi Dengan Metode Nth (Studi Kasus: Laboratorium Teknik Informatika Institut Teknologi Padang). *Teknoif*, *5*(1), 56–62. https://doi.org/10.21063/JTIF.2017.V5. 1. 56-62

[15] Yusuf, E., Riza, T. A., Ariefianto, T., & Elektro, F. (2013). Implementasi Teknologi Load Balancer Dengan Web Server Nginx Untuk Mengatasi Beban Server. *Seminar Nasional Teknologii Informasi Dan Multimedia*, 11–16.