InComTech: Jurnal Telekomunikasi dan Komputer
vol.10, no.2, Agustus 2020, 135-144
http://publikasi.mercubuana.ac.id/index.php/Incomtech
P-ISSN: 2085-4811 E-ISSN: 2579-6089

Analisis Perbandingan Server *Load Balancing* dengan Haproxy & Nginx dalam Mendukung Kinerja Server E- Learning

Sampurna Dadi Riskiono[1*] Donaya Pasha[2]

[1]*Teknik Elektro, Fakultas Teknik dan Ilmu Komputer, Universitas Teknokrat Indonesia*
[2]*Teknologi Informasi, Fakultas Teknik dan Ilmu Komputer, Universitas Teknokrat Indonesia*
Jl. Z. A. Pagar Alam No.9 -11, Kota Bandar Lampung 35132, Indonesia
*E-mail Penulis Koresponden: sampurna.go@teknokrat.ac.id

Abstrak
Perkembangan dunia internet terus meningkat. Hal ini terlihat dengan berbagai sistem yang dapat diakses secara online, termasuk implementasi sistem *e- learning*. Kondisi ini harus didukung dengan infrastruktur server yang memadai. Server harus mampu menangani lonjakan *request* yang dilakukan setiap user. Oleh karenanya perlu dibangun sistem server dengan memanfaatkan metode *load balancing*. Sistem *load balancing* mampu mendistribusikan beban secara merata terhadap masing-masing server yang ada dilingkungannya. Untuk itu perlu dilakukan pengujian terhadap *server load balancing* yang akan menjadi tumpuan *request* user sebelum distribusikan ke masing-masing server. Penelitian ini akan melakukan pengujian terhadap Haproxy dan Nginx dengan melakukan pengukuran terhadap dua variable yakni *throughput* dan *response time*. Hasil pengujian menunjukkan implementasi *load balancing* dengan Haproxy memiliki nilai *response time* sebesar 585 ms lebih kecil dengan Nginx yang memiliki waktu *response time* 1180,58 ms pada uji koneksi 300/300 sec. Sedangkan untuk nilai *throughput load balancing* dengan Haproxy juga menunjukkan kinerja yang lebih baik dengan nilai sebesar 896,48 Kb/s lebih besar dibandingkan dengan Nginx yang berada pada angka 848,52 Kb/s pada uji koneksi 300/300 sec. Berdasarkan pengujian yang telah dilakukan, penerapan *load balancing* dengan Haproxy lebih baik dibandingkan dengan Nginx.

Keywords:
E-Learning;
Haproxy*;*
Load balancing;
Nginx*;*

Article history:
Diserahkan 11 Juli 2020
Direvisi 4 Oktober 2020
Diterima 9 Oktober 2020
Dipublikasi 9 Desember 2020

DOI:
10.22441/incomtech.v10i3.8751

1. PENDAHULUAN

Sejalan dengan meningkatnya kompleksitas layanan dan aplikasi web dalam berbagai bidang, maka permintaan layanan web dari pengguna semakin

meningkat. Contoh layanan dan aplikasi web yang populer adalah layanan dan aplikasi bisnis (*e-business*), pendidikan (*e-learning*), berita (*news*), dan lain-lain [1, 2, 3, 4]. Dalam meningkatkan sistem layanan *e-learning*, dibutuhkan suatu sistem server yang dapat mengatasi sejumlah akses yang tinggi. Hal ini terkait dengan skalabilitas sistem, dimana model pelayanan server jamak menjadi sebuah pilihan dengan menerapkan metode *load balancing*.

Load balancing merupakan salah satu model yang dapat digunakan untuk meningkatkan kinerja dan ketersediaan server, yaitu dengan mendistribusikan permintaan layanan yang datang ke beberapa server sekaligus, sehingga beban yang diterima oleh masing-masing server lebih sedikit [5, 6, 7]. Dalam mengatasi peningkatan kinerja ketika adanya akses yang begitu banyak, maka penggunaan banyak server dapat menjadi solusi untuk mengatasi hal tersebut. Himpunan dari banyak server disebut dengan kluster server. Sehingga dengan menerapkan kluster server maka dapat meningkatkan keandalan dan ketersediaan aplikasi [8].

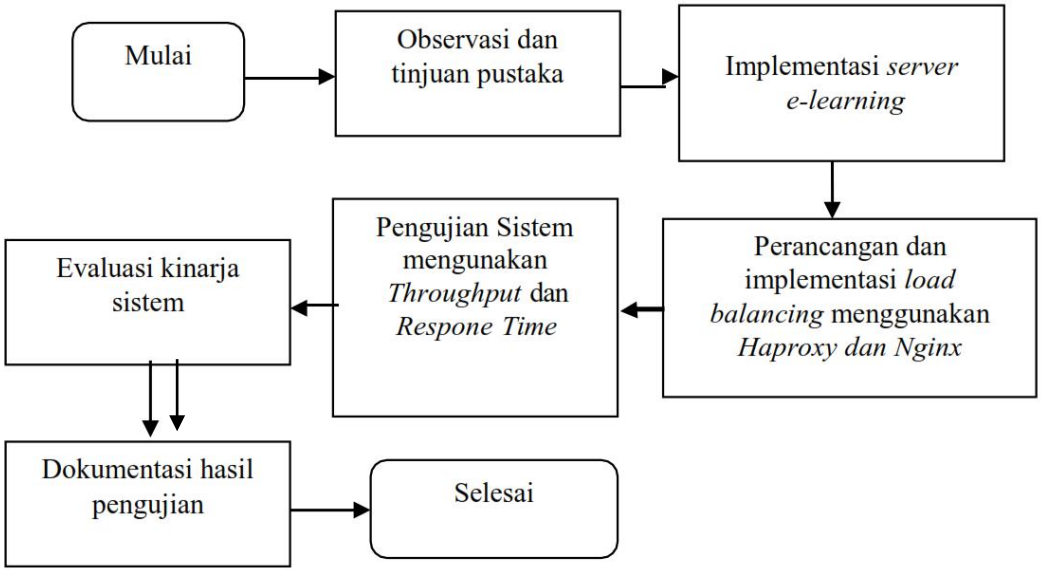
Server *load balancing* memiliki tugas dalam mendistribusikan beban kerja ke banyak server dengan mempertimbangkan kapasitas dari setiap server yang ada sehingga dapat mengurangi terjadinya kegagalan server [9]. Dengan penerapan banyak server, maka ketika terjadi kegagalan pada salah satu server, layanan yang ada masih terus dapat berjalan. Hal lain adalah beban yang semakin kecil, karena beban yang ada kemudian didistribusikan ke masing-masing server sehingga sistem yang ada menjadi terdistribusi. Sistem terdistribusi menyediakan berbagai sumberdaya, sebagai salah satu keuntungan utamanya adalah dapat menyediakan kinerja yang lebih baik serta keandalan dari pada sistem tradisional yang lain dalam kondisi yang sama [10].

Penggunaan banyak server untuk sistem terdistribusi membutuhkan metode untuk mengatur pembagian beban secara adil atau merata pada masing-masing server. Banyak penelitian telah dilakukan terkait penerapan metode *load balancing* untuk mengatur pembagian beban kerja pada server dengan tujuan untuk meningkatkan kinerja sistem. Penerapan *load balancing* dalam web server sangat penting dan dapat merupakan sebuah solusi yang tepat dan efektif untuk menangani beban server yang sibuk serta diharapkan dapat meningkatkan skalabilitas pada sistem terdistribusi [11]. Web server harus bebas dari kegagalan baik akibat dari hardware ataupun software. Dalam kondisi yang lain fault tolerance dapat diterapkan untuk mendeteksi dan mentoleransi kerusakan secara real-time pada sistem terdistribusi [12].

2. METODE

Dalam penelitian ini, metode *load balancing* digunakan untuk meningkatkan kinerja dari server *e-learning*dengan membandingkan kondisi *load balancing* yang didukung dengan HaProxy dan *load balancing* yang didukung dengan Nginx. Untuk tahap awal akan dilakukan tinjauan pustaka dan observasi. Proses selanjutnya adalah implemantasi sistem *e-learning*dan dilanjutkan dengan implementasi dari metode *load balancing*. Pengujian dilakukan dengan cara memberikan sejumlah koneksi terhadap server e-learning, baik *load balancing* yang didukung dengan HaProxy maupun *load balancing* yang didukung dengan Nginx. Selanjutnya dilakukan evaluasi terhadap implementasi metode *load balancing* tersebut dengan menggunkan parameter pengujian yang telah

ditentukan. Dari hasil evaluasi, maka terlihat nilai hasil dari uji parameter yang telah dilakukan. Untuk memperjelas jalannya penelitian ini secara lebih detail metode yang akan dilakukan ditunjukkan pada Gambar 1.



Gambar1. Metode Penelitian

Adapun yang menjadi landasan terori dari penelitian ini adalah sebagai berikut.

2.1. *E-Learning*

E-learning merupakan kependekan dari *Electronic Learning* adalah sebuah proses pembelajaran dimana penyampaian materi, diskusi, ujian dan lain-lain yang berkaitan dengan kegiatan perkuliahan dilakukan melalui media elektronik [13]. Dari beberapa sistem *e-learning* yang ada, secara umum dapat dibagi berdasarkan sifat interaktivitasnya dan dapat dibedakan ke dalam dua kelompok yaitu *e-learning* statis dan *e-learning* dinamis. Sistem *e-learning* dikatakan bersifat statis jika antara pengguna sistem tidak dapat saling berinteraksi, pembelajar hanya dapat melakukan proses download bahan-bahan yang diperlukan dan admin hanya dapat melaukan proses upload file-file materi. Sistem ini biasanya digunakan hanya sebagai penunjang aktifitas belajar-mengajar yang dilakukan secara tatap muka di kelas. Sedangkan sistem *e- learning* dapat digolongkan kedalam *e-learning* yang bersifat dinamis apabila siswa mampu belajar dengan dalam lingkungan yang tidak jauh berbeda dengan suasana kelas dimana di dalam sistem ini terdapat kemungkinan untuk berinteraksi antara pembelajar dan tutornya baik melalui email, chatting maupun sarana komunikasi lainnya [14].

2.2. *Load balancing*

Load balancing merupakan metodologi jaringan komputer yang bekerja dengan cara mendistribusikan beban kerja dibeberapa komputer atau kluster komputer untuk mencapai pemamfaatan optimal dari sumber daya, memaksimalkan *throughput*, meminimalkan waktu respon, dan menghindari kelebihan beban. Kluster server sendiri merupakan gabungan beberapa perangkat komputer yang saling terhubung dan bekerjasama sehingga mereka dapat dilihat sebagai satu sistem dalam banyak aspek, dan kluster komputer biasanya digunakan untuk

meningkatkan kinerja dan ketersediaan atas satu komputer [11]. *Load balancer* dapat melakukan berbagai fungsi seperti peyeimbang beban, *traffic engineering*, dan pemindahan jalur trafik. *Load balancer* dapat melakukan pemeriksaan kesehatan pada server, aplikasi, dan konten untuk meningkatkan ketersediaan layanan dan pengelolaanya [15]. Dalam penerapan *load balancer* ada berbagai algoritma penjadwalan yang dapat digunakan untuk mendistribusikan beban kepada setiap server didalam sekumpulan server. Algoritma penjadwalan yang umumnya digunakan adalah: *Round robin* dan *Least connection.*

Round robin

Untuk model penjadwalan tipe round robin, *load balancer* mendistribusikan permintaan layanan sama rata ke seluruh real server tanpa memperdulikan kapasitas server atau pun beban permintaan layanan. Jika ada empat real server (A,B,C,D), maka permintaan layanan 1 akan diberikan *load balancer* kepada server A, permintaan layanan 2 ke server B, permintaan layanan 3 ke server C, permintaan layanan 4 ke server D dan permintaan layanan 4 akan kembali ke server A. Mekanisme ini dapat dilakukan jika seluruh real server menggunakan spesifikasi komputer yang sama. Konsep dasar dari algoritma ini adalah dengan menggunakan time-sharing. Pada dasarnya algoritma ini sama dengan FCFS, hanya saja bersifat preemptive. Setiap proses mendapatkan waktu CPU yang disebut dengan waktu quantum (*quantum time*) untuk membatasi waktu proses, biasanya 1-100 milidetik. Setelah waktu habis, proses ditunda dan ditambahkan pada *ready queue*. Algoritma *roundrobin* merupakan algoritma yang paling sederhana dan banyak digunakan oleh perangkat *load balancing*. Algoritma ini membagi beban secara bergiliran dan berurutan dari satu server ke server lain sehingga membentuk putaran. Penjadwalan ini merupakan:

1. Penjadwalan preemptive, bukan di-preempt oleh proses lain, tapi terutama oleh penjadwal berdasarkan lama waktu berjalannya proses, disebut *preempt by time*.
2. Penjadwal tanpa prioritas. Semua proses dianggap penting dan diberi sejumlah waktu proses yang disebut kwanta (quantum) atau time slice dimana proses itu berjalan.

Ketentuan algoritma *round robin* adalah sebagai berikut:

1. Jika kwanta dan proses belum selesai maka proses menjadi *runnable* dan pemroses dialihkan ke proses lain.
2. Jika kwanta belum habis dan proses menunggu suatu kejadian (selesainya operasi I/O), maka proses menjadi *blocked* dan pemroses dialihkan ke proses lain.
3. Jika kwanta belum habis tapi proses telah selesai, maka proses diakhiri dan pemroses dialihkan ke proses lain. Algoritma *round robin* menggilir proses yang ada di antrian. Proses akan mendapat jatah sebesar quantum time. Jika quantum time-nya habis atau proses sudah selesai, CPU akan dialokasikan ke proses berikutnya. Tentu proses ini cukup adil karena tak ada proses yang diprioritaskan, semua proses mendapat jatah waktu yang sama dari CPU yaitu (1/n), dan tak akan menunggu lebih lama dari (n-1)q dengan q adalah lama 1 quantum. Algoritma ini sepenuhnya bergantung besarnya *time quantum*. Jika terlalu besar, algoritma ini akan sama saja dengan algoritma *first come first served*. Jika terlalu kecil, akan semakin banyak peralihan proses sehingga

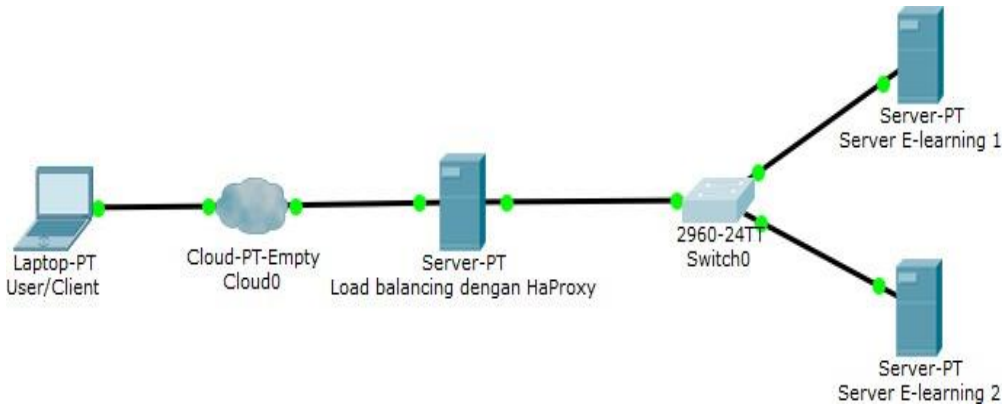
banyak waktu terbuang. Permasalahan utama pada round robin adalah menentukan besarnya time quantum. Jika time quantum yang ditentukan terlalu kecil, maka sebagian besar proses tidak akan selesai dalam 1 quantum. Hal ini tidak baik karena akan terjadi banyak switch, padahal CPU memerlukan waktu untuk beralih dari suatu proses ke proses lain (disebut dengan context switches time). Sebaliknya, jika time quantum terlalu besar, algoritma Roundrobin akan berjalan seperti algoritma *first come first served*. Time quantum yang ideal adalah jika 80% dari total proses memiliki CPU burst time yang lebih kecil dari 1 time quantum [16].

Least connection

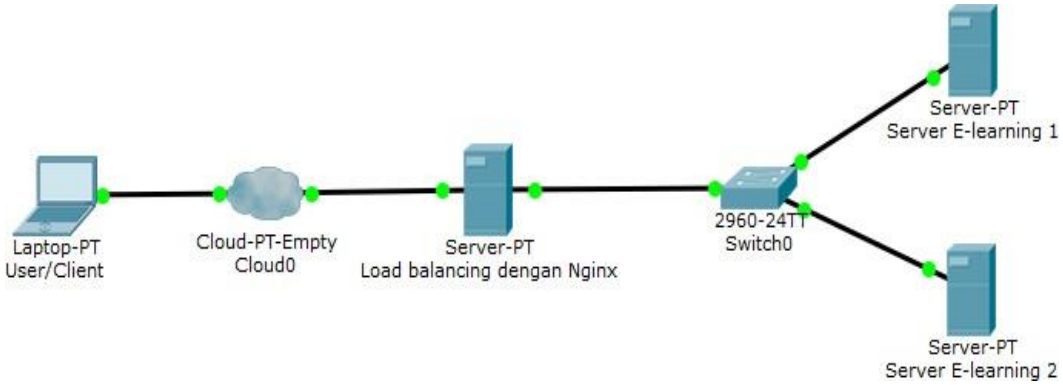
Algoritma ini melibatkan pengiriman permintaan layanan baru ke server dengan jumlah koneksi bersamaan. Metode ini memerlukan penyeimbang beban untuk melacak jumlah koneksi aktif secara bersamaan pada setiap server dan mengirim permintaan ke server dengan sedikitnya jumlah koneksi yang aktif bersamaan [16].

3. HASIL DAN PEMBAHASAN

Gambar 2 menjelaskan bagaimana bentuk dari arsitektur Server *Load balancing* dengan HaProxy yang mana terdiri dari tiga unit server. Server *Load balancing* dengan HaProxy, App server1 *e-learning*, App server2 *e-learning* terletak pada jaringan cloud. Server *load balancing* ini yang berfungsi dalam mendistribusikan beban koneksi yang dilakukan oleh user. Nantinya koneksi yang ada akan didistribusikan merata ke masing-masing server *e-learning*.



Gambar 2. Arsitektur Server *Load balancing* dengan HaProxy



Gambar 3. Arsitektur Server *Load balancing* dengan Nginx

Gambar 3 menunjukkan bahwa *Server Load balancing* dengan Nginx, App server1 e-learning, App server2 *e-learning*terletak pada jaringan cloud. Sedangkan client dapat melakukan akses melalui jaringan publik. Untuk detail dari antarmuka jaringan yang terpasang akan diperlihatkan pada Tabel 1.

Tabel 1. Nama Server dan IP Address

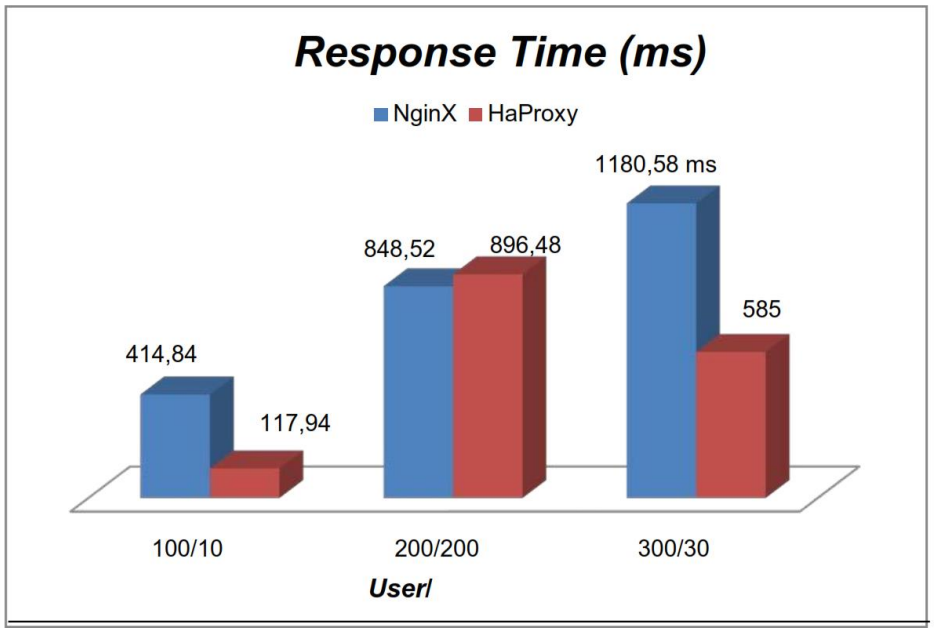
Nama Server	IP Address
Server *Load balancer* HaProxy/Nginx	103.41.205.217
Server 1 E-Learning	103.41.204.218
Server 1 E-Learning	103.41.205.204
User/Client	192.168.7.131

Rincian pada Tabel 1 menunjukkan bahwa alamat IP 103.41.205.217 merupakan alamat yang dimiliki oleh *load balancer* yang berfungsi untuk meneruskan dan mendistribusikan permintaan ke aplikasi server. Ketika pengguna meminta konten di server aplikasi *e-learning*, maka semua permintaan akan melewati server *load balancer*. Jadi pengguna tidak akan mengetahui *real server* mana yang melayani. Kemudian permintaan layanan tersebut akan dibagikan oleh server *load balancer* kepada *real server* yang terdaftar pada *load balancer* dengan menyesuaikan pada algoritma penjadwalan yang telah ditanamkan. Pengamatan dalam penelitian ini dilakukan dengan mengambil dua parameter yaitu *throughput* dan *response time*. Pengujian ini bertujuan untuk melakukan evaluasi kinerja dengan diimplementasikannya metode *load balancing*. Dalam hal ini yang mana pada sisi klien akan dilakukan permintaan secara bertahap melalui tool httperf untuk melihat *throughput* dan *response time*. Untuk lebih jelasnya terlihat pada Tabel 2 yang merupkan hasil rata-rata dari permintaan layanan yang di lakukan oleh *client* melalui tool httperf. Dimana permintaan layanan tersebut akan dilakukan secara bertahap. Dimulai dari jumlah koneksi 100/100 sec, 200/200 sec, dan 300/300 sec yang dilakukan koneksi ke server secara bersamaan dalam satu waktu. Dari sini maka dapat dilakukan pengamatan serta membandikan kinerja dari kedua model yang telah di implementasikan. Berdasarkan pada hasil yang telah diperoleh dari nilai *response time* dan nilai *throughput* seperti yang di perlihatkan pada Tabel 2.

Tabel 2. Perbandingan uji koneksi Nginx dan HaProxy

User/Koneksi	Nginx		HaProxy	
	Throughput (KB/S)	*Respon Time (ms)*	*Throughput* (KB/S)	*Respone Time (ms)*
100/100	20,84	414,84	49,64	117,94
200/200	36,46	848,52	33,14	896,48
300/300	31,62	1180,58	39,76	585
Rata-rata	29,64	814,6466667	40,84666667	533,14

Gambar 4 memperlihatkan perbandingan secara grafik antara *load balancing* dengan Nginx dan *load balancing* dengan HaProxy dari nilai *response time* yang telah diukur.

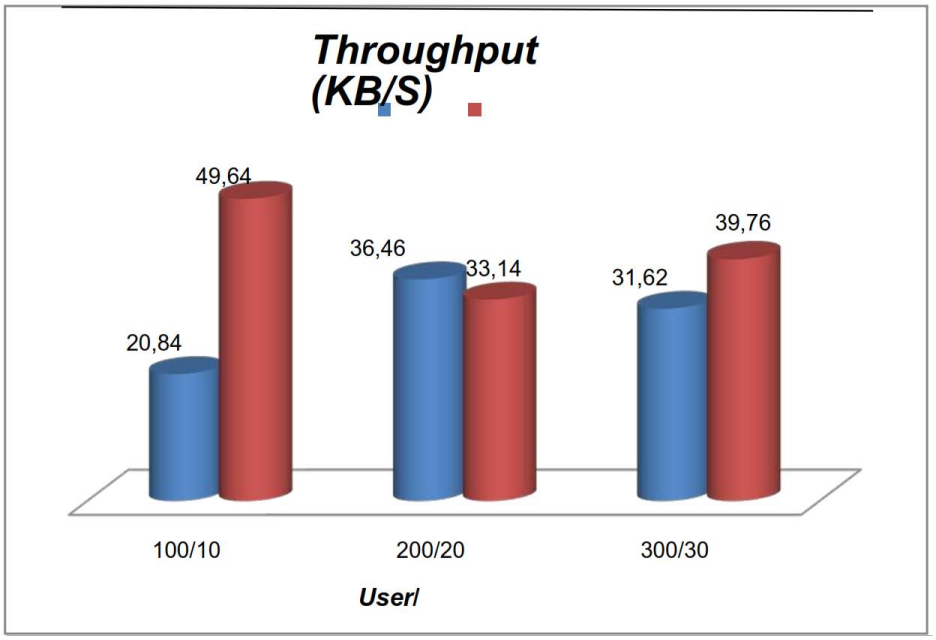


Gambar 4. Grafik *Response time* (ms)

Hasil dari perbandingan kedua model arsitektur antara *load balancing* dengan Nginx dan *load balancing* dengan HaProxy dapat dilihat pada Tabel 2, dimana penerapan *load balancing* dangan HaProxy memiliki nilai *response time* lebih kecil di bandingkan dengan penerapan *load balancing* dengan Nginx dari setiap rentang pengujinnya. Pada uji koneksi 100/100 *sec nilai reponse time* dari *load balancing* dangan HaProxy adalah 117,94 ms, lebih kecil dibandingkan penerapan *load balancing* dengan Nginx yang memiliki nilai *response time* sebesar 414,84 ms. Ini menujukan bahwa kecepatan *response time* yang didukung penerapan *load balancing* dengan HaProxy lebih cepat dibandingkan penggunaan *load balancing* dengan Nginx. Sehingga dari segi layananan, penerapan *load balancing* dengan HaProxy lebih layak untuk diimplemantasikan karena ini merupakan bagian dari *Quality of Service* (QoS) dari suatu layanan. Hal serupa juga terjadi pada uji koneksi selanjutnya, dimana pada uji koneksi 200/200 *sec*, dan 300/300 *sec* penerapan server jamak yang didukung metode *load balancing* dengan HaProxy lebih unggul bila dibandingkan dengan penerapan *load balancing* dengan Nginx. Bahkan untuk nilai koneksi 300/300 *sec*, perbedaan *respone time* sangat besar, dimana *load balancing* dengan HaProxy memiliki nilai 585 ms sedangkan *load*

balancing dengan Nginx sebesar 1180,58 ms. Namun untuk uji koneksi 200/200 sec nilai *response time load balancing* dengan HaProxy lebih besar di bandingkan *load balancing* dengan Nginx.

Hal serupa juga terjadi pada saat pengujian untuk mendapatkan nilai *throughput*. Hasil dari perbandingan kedua model arsitektur antara *load balancing* dengan Nginx dan *load balancing* dengan HaProxy dapat dilihat pada Tabel 2, dimana penerapan *load balancing* dangan HaProxy memiliki nilai *throughput* lebih besar di bandingkan dengan penerapan *load balancing* dengan Nginx dari setiap rentang pengujinnya. Pada uji koneksi 100/100 sec nilai *throughput* dari *load balancing* dangan HaProxy adalah 49,64 Kb/sec, lebih besar dibandingkan penerapan *load balancing* dengan Nginx yang memiliki nilai *throughput* sebesar 20,84 Kb/sec. Ini menujukan bahwa besaran nilai *throughput* yang didukung penerapan *load balancing* dengan HaProxy lebih besar dibandingkan penggunaan *load balancing* dengan Nginx. Sehingga dari segi layananan, penerapan *load balancing* dengan HaProxy lebih layak untuk diimplemantasikan karena ini merupakan bagian dari quality of service (QoS) dari suatu layanan. Hal serupa juga terjadi pada uji koneksi selanjutnya, dimana pada uji koneksi 200/200 sec, dan 300/300 sec penerapan server jamak yang didukung metode *load balancing* dengan HaProxy lebih baik bila dibandingkan dengan penerapan *load balancing* dengan Nginx. Untuk nilai koneksi 300/300 sec, perbedaan *throughput load balancing* dengan HaProxy juga lebih besar, dimana *load balancing* dengan HaProxy memiliki nilai 39,76 Kb/sec sedangkan *load balancing* dengan Nginx sebesar 31,62 Kb/sec. Namun untuk uji koneksi 200/200 sec nilai *troughput load balancing* dengan HaProxy lebih kecil di bandingkan *load balancing* dengan Nginx.



Gambar 5. Hasil uji koneksi terhadap server tunggal

Gambar 5 memperlihatkan perbandingan secara grafik antara *load balancing* dengan Nginx dan *load balancing* dengan HaProxy dari nilai *throughput* yang telah diukur.

Dari hasil total rata-rata pengujian, implemantasi *load balancing* dengan HaProxy lebih unggul dibandingkan *load balancing* dengan Nginx. Dalam penelitian ini, *load balancer* masih bersifat tunggal. Belum dikembangkan bagaimana membangun sebuah sistem yang memiliki ketersediaan yang tinggi dengan menerapkan sistem yang *fault tolerance.* Dengan sistem yang *fault tolerance* diharapkan ketersediaan sistem akan terus terjaga.

4. KESIMPULAN

Dari hasil pengujian yang telah dilakukan maka dapat diambil kesimpulan. Hasil perbandingan nilai rata-rata menunjukkan *load balancing* dengan HaProxy memiliki nilai *response time* 533,14 ms lebih kecil dibandingkan dengan *load balancing* dengan Nginx yang memiliki nilai *response time* 814,64 ms. Ini menunjukkan waktu respon *load balancing* dengan HaProxy lebih cepat dibanding *load balancing* dengan Nginx. Untuk *throughput*, *load balancing* dengan HaProxy juga memiliki nilai yang lebih besar yaitu 40,84 Kb/sec dibandingkan *load balancing* dengan Nginx yang memiliki nilai 29,64 Kb/sec. Ini menunjukkan bahwa jumlah paket yang dapat dilewatkan *load balancing* dengan HaProxy lebih besar dari pada *load balancing* dengan Nginx. Maka, berdasarkan pengujian yang telah dilakukan, menunjukkan bahwa penerapan *load balancing* dengan HaProxy dalam meningkatkan kinerja telah berhasil diimplementasikan. Dengan melihat nilai *response time* yang lebih kecil dibanding dengan penerapan *load balancing* dengan Nginx dan juga nilai troughput yang lebih besar.

UCAPAN TERIMAKASIH

Ucapan TerimakasihTerima kasih kepada Direktorat Riset dan Pengabdian kepada Masyarakat (DRPM) Dikti yang telah mendanai kegiatan penelitian ini, yakni pada skema Penelitian Dosen Pemula (PDP) sesuai dengan SK Penetapan Pemenang Hibah PDP nomor: T/140/E3/RA.00/2019 tanggal 28 Februari 2019 dan Kontrak Pelaksanaan Penelitian Nomor: 010/LPPM-UTI/FTIK/PDP-MONO/V/2019 tanggal 2 Mei 2019. Terima kasih juga Peneliti sampaikan kepada LPPM Universitas Teknokrat Indonesia yang telah memfasilitasi kegiatan penelitian ini.

REFERENSI

[1] N. Angsar, “Pengujian Distribusi Beban Web dengan Algoritma Least Connection dan Weighted Least Connection,” *JNTETI*, vol. 3, no. 1, pp. 24–28, 2014, DOI: 10.22146/jnteti.v3i1.38

[2] M. Jazayeri, "Some Trends in Web Application Development," Future of Software Engineering (FOSE '07), Minneapolis, MN, 2007, pp. 199-213, DOI: 10.1109/FOSE.2007.26.

[3] H. Huang, Z. Zhang, H. Cheng and S. W. Shieh, "Web Application Security: Threats, Countermeasures, and Pitfalls," in Computer, vol. 50, no. 6, pp. 81-85, 2017, DOI: 10.1109/MC.2017.183.

[4] S. M. Murshed, A.M. Al-Hyari, J. Wendel, L. Ansart, “Design and Implementation of a 4D Web Application for Analytical Visualization of Smart City Applications,” *ISPRS Int. J. Geo-Inf,* vol. 7, pp. 276, 2018, DOI: 10.3390/ijhi7070276

[5] D. Lukitasari, F. Oklilas, F. I. Komputer, and U. Sriwijaya, “Analisis Perbandingan *Load balancing* Web Server Tunggal dengan Web server Cluster Menggunakan Linux Virtual

Server," *Jurnal Generik,* vol. 5, no. 2, pp. 31–34, Juli 2010.
[6] M. K. Anwar dan I. Nurhaida, "Implementasi Load Balancing menggunakan Metode Equal Cost Multi Path (ECMP) pada Interkoneksi Jaringan," *InComTech: Jurnal Telekomunikasi dan Komputer,* vol. 19, no. 1, pp. 39-48, April 2019, DOI: 10.2241/incomtech,v9i1.5962
[7] E. J. Ghomi, A. M. Rahmani, dan N. N. Qader,"Load-balancing algorithms in cloud computing: A survey," *Journal of Network and Computer Applications*,vol. 88, pp. 50-71,2017, DOI: 10.1016/j.jnca.2017.04.007.
[8] S. D. Riskiono, S. Sulistyo, and T. B. Adji, "Kinerja Metode *Load balancing* dan Fault Tolerance Pada Server Aplikasi Chat," Prosiding Seminar Nasional ReTII ke-11, 2017.
[9] C. Kopparapu, *Load balancing Servers, Firewalls, and Caches*, Chandra Kopparapu, 2002
[10] J. A. Rabu, J. Purwadi dan S. Raharjo, "Implementasi *Load balancing* Web Server Menggunakan Metode LVS-NAT," *Jurnal Informatika,* vol. 8, no. 2, 2012, DOI: 10.21460/inf.2012.82.126
[11] A. Alimuddin dan A. Ashari, "Peningkatan Kinerja Siakad Menggunakan Metode *Load balancing* dan Fault Tolerance di Jaringan Kampus Universitas Halu Oleo," *IJCCS (Indonesian Journal of Computing and Cybernetics Systems),* vol. 10, no. 1, pp. 11–22, 2016.
[12] H. T. Chang, "*Load balancing* and Fault-Tolerance for Scalable Network File Systems Using by Web Services," Proceedings *of the WSEAES 13th international conference on Computers*, July 2002, pp. 351–356.
[13] A. Ambarita, "Implementasi Sistem *E-learning* Menggunakan Software Moodle Pada Politeknik Sains dan Teknologi Wiratama Maluku Utara," *IJIS - Indones. J. Inf. Syst*., vol. 1, no. 2, 2017, DOI: 10.36549/ijis.v1i2.17
[14] B. Suteja dan A. Harjoko, "User Interface Design for *e-learning*," *Seminar Nasional Aplikasi Teknologi Informasi 2008 (SNATI 2008),* Yogyakarta, Indonesia, 2008, pp. 1-10.
[15] S. Malik, "Dynamic *load balancing* in a network of workstations," *Pap. Parallel Process*. Course, Carlet., no. 219762, 2000.
[16] G. Triono, "Implementasi *Load Balancing* Dengan Menggunakan Algoritma Round Robin Pada Kasus Pendaftaran Siswa Baru Sekolah Menengah Pertama Labschool Unesa Surabaya," Seminar Nasional "Inovasi dalam Desain dan Teknologi" - IDeaTech 2015, pp. 169–176