e-ISSN 2962-6129
Seminar Nasional Mahasiswa Ilmu Komputer dan Aplikasinya (SENAMIKA)
Jakarta-Indonesia, 20 Agustus 2022

Penerapan *Cloud Load Balancing* Dengan menggunakan HAProxy Dalam Meningkatkan *Server Availability* Pada Studi Kasus *Learning Management System* (LMS) Universitas XYZ

Ariansyah Arifin[1], Gito Putro Wardana[2], Syamsul Arifin[3], M Fadlan Ridho[4], Hamonangan Kinantan Prabu[5]
Program Studi Informatika / Fakultas Ilmu Komputer
Universitas Pembangunan Nasional Veteran Jakarta
Jl. RS. Fatmawati, Pondok Labu, Jakarta Selatan, DKI Jakarta, 12460, Indonesia.
ariansyaha@upnvj.ac.id[1], gito@upnvj.ac.id[2], syamsul@upnvj.ac.id[3], mfadlan@upnvj.ac.id[4],
hamonangan.kp@upnvj.ac.id[5]

Abstrak. Dunia pendidikan saat ini berkembang sangat cepat, hal ini ditandai oleh sebuah pemanfaatan teknologi informasi dalam pendidikan. Ketika jumlah pengguna yang mengakses layanan *Learning Management System* di Universitas XYZ meningkat dari berbagai fakultas dan *server* tidak bisa menangani layanan beban koneksi tersebut atau *server* yang digunakan hanya satu serta mengalami *down*, tentu akan menjadi masalah. Salah satu cara untuk mengatasi masalah ini adalah dengan mengimplementasikan sistem arsitektur *server* berbasis *cloud* dengan metode *load balancing* HAproxy. Dalam penelitian ini memiliki tujuan yaitu akan dilakukan evaluasi antara penggunaan satu *server* lokal dan yang menggunakan *server* berbasis *cloud* dengan *load balancing* HAproxy dalam meningkatkan *availability* di *Learning Management System* Universitas XYZ dengan pengujian melakukan pengukuran terhadap tiga variabel yakni *response time, request concurrent* dan *failover*. Hasil pengujian menunjukkan implementasi *server cloud clustering load balancing* dengan HAproxy lebih baik dari segi *response time, request concurrent,* dan sistem *failover* dibanding dengan hasil dari *server* tunggal lokal.

Kata Kunci. *server Learning Management System* Universitas XYZ, Beban Koneksi, *availability, cloud load balancing,* HAproxy, *response time, request concurrent,* sistem *failover* .

1 Pendahuluan

Seiring dengan perkembangan dunia teknologi dalam pendidikan yang sangat pesat, hal tersebut ditandai dengan sebuah pemanfaatan teknologi informasi sebagai penunjangnya. Penggunaan teknologi informasi ini menyebabkan proses belajar mengajar menjadi lebih menarik dan efektif. Pada awalnya pemanfaatan teknologi informasi hanya sebatas penyampaian materi presentasi menggunakan Power Point, Adobe Flash, atau aplikasi khusus lainnya yang memiliki fungsi sama. Namun seiring dengan pertumbuhan teknologi internet dan di masa pandemi ini, pembelajaran melalui teknologi semakin masif, proses belajar mengajar semakin banyak menggunakan aplikasi berbasis internet, termasuk penggunaan *Learning Management System* (LMS) di Universitas XYZ yang bisa melakukan pembelajaran berupa tempat menyediakan materi, *quiz*, forum, interaksi langsung dalam *Learning Management System* (LMS) antar pengguna, dan media informasi yang efektif untuk mendukung kegiatan belajar jarak jauh para mahasiswa/i.

Sejalan dengan semakin kompleksnya layanan *Learning Management System* (LMS) di Universitas XYZ pada berbagai sektor, permintaan layanan web *Learning Management System* (LMS) dari pengguna semakin meningkat. Setiap bisnis atau organisasi pemerintah yang memelihara ribuan atau bahkan jutaan data setiap hari menghadapi kesulitan menemukan infrastruktur teknologi informasi yang solid dan dapat diandalkan. Server merupakan salah satu infrastruktur yang digunakan untuk pengelolaan data [1]. Untuk meningkatkan sistem layanan *Learning Management System* (LMS) diperlukan sistem *server* yang dapat mengelola akses dalam jumlah besar. Ini berkaitan dengan skalabilitas dan ketersediaan sistem, jadi untuk saat ini, model layanan multi-*server* dengan penggunaan HAproxy sebagai mekanisme *load balancing* adalah opsi yang lebih disukai. Dengan mendistribusikan beban di antara semua server di cluster, *load balancing* adalah teknik untuk meningkatkan kinerja *server*. Permintaan koneksi HTTP yang dikirim ke web *server* akan diproses oleh *server* dengan keadaan terbaik berkat mekanisme *load balancing*, dimana pemilihan dilakukan dengan menggunakan sistem prioritas web *server* [2].

Beberapa penelitian terdahulu terkait sistem *load balancing* pernah dilakukan seperti peran *Load Balancing* Dalam Meningkatkan Kinerja *Web Server* di Lingkungan *Cloud* dengan sebuah hasil pengujian pada target web server testing default yang dibuat di dapatkan *response time* 61.19 *ms* yang lebih kecil dari tanpa load balancing [3]. Lalu ada Perancangan Dan Implementasi *Load Balancing Reverse Proxy* Menggunakan HaProxy Pada Aplikasi Web

dengan hasil pengujian aplikasi web upload-download yang di pasang load balancing berhasil membagi beban request dari pengguna ke kedua server [4]. Namun selama ini penerapan terkait peningkatan *server availability* pada studi kasus *Learning Management System* (LMS) dengan HaProxy *Load Balancing Cloud* yang masih kurang diterapkan sehingga bisa berdampak pada aktivitas pada LMS yang tidak menggunakan sistem *load balancing*. Maka dari itu sistem *cloud load balancing* dengan menggunakan haproxy dalam meningkatkan *server availability* pada studi kasus *Learning Management System* (LMS) ini sangat penting.

2 Landasan Teori

2.1 *Learning Management System* (LMS)

LMS atau *Learning Management System* merupakan *software* atau perangkat lunak yang digunakan untuk pencarian materi, dokumentasi, administrasi, pelaporan suatu kegiatan, penyediaan materi pelatihan untuk kegiatan belajar mengajar secara *online* yang terhubung dengan internet [5]. LMS sendiri menyediakan berbagai fitur yang memberikan layanan untuk mempermudah dalam mengunggah dan berbagi bahan ajar, forum diskusi *online*, *chat* antar pengguna, *live quiz*, *survey*, laporan nilai, dan lain sebagainya. Dengan pertumbuhan LMS yang signifikan, pengajar serta mahasiswa/i mencari cara baru buat membentuk pengetahuan, meningkatkan pengajaran dan pengalaman di kelas secara *online* tanpa interaksi tatap muka, sementara pada beberapa konteks program yang ditawarkan merupakan mode campuran dalam bentuk *hybrid* yang artinya penggunaan keduanya secara tatap muka serta interaksi *online* yang difasilitasi oleh teknologi informasi pendidikan. Lingkungan belajar *online* bisa memberikan pengajar, mahasiswa/i kesempatan untuk fleksibilitas, kemudahan, dan kolaborasi [6].

2.2 Web Server

Web server bisa merujuk baik ke perangkat keras maupun perangkat lunak yang menyediakan layanan berbasis data menggunakan protokol HTTP atau HTTPS. Pada umumnya fungsi *web server* adalah untuk meletakkan situs web. Dalam mekanismenya dari klien menggunakan aplikasi *web browser* untuk meminta data dan *server* akan memindahkan data dalam bentuk halaman web serta umumnya dalam bentuk dokumen HTML. Halaman web yang diminta dapat terdiri dari file teks, video, gambar, *file*, dan lainnya. Sebuah standar supaya web *server* bisa dikatakan berjalan dengan proper adalah terkoneksi sebuah komunikasi *request*, *transfer* dan menerima, contohnya dengan protokol HTTP/HTTPS dari *server* ke *client* atau sebaliknya tanpa ada paket data yang hilang ataupun tidak sampai dalam proses *request*, *transfer*, dan menerima [2]. Salah satu program *web server* adalah Apache. Apache adalah *server* web yang paling banyak digunakan pada internet. program ini awalnya dirancang buat lingkungan sistem operasi UNIX, namun kini apache tersedia, yang dirancang buat sistem operasi lain. Apache mempunyai sejumlah besar program *support*. Ini memberi pengguna layanan yang relatif lengkap [7].

2.3 Jaringan Komputer

Jaringan komputer adalah sistem media komunikasi, perangkat keras, dan perangkat lunak yang diperlukan untuk menghubungkan dua atau lebih sistem dan perangkat komputer yang saling berhubungan satu sama lain [8]. Ada lima jenis jaringan komputer, yaitu berdasarkan wilayah geografis, media transmisi data, distribusi informasi/sumber data, peranan dan hubungan masing-masing komputer dalam pemrosesan data, serta jenis topologi yang digunakan. Itu adalah semua faktor yang perlu dipertimbangkan. Jenis jaringan komputer berbasis geografis meliputi:

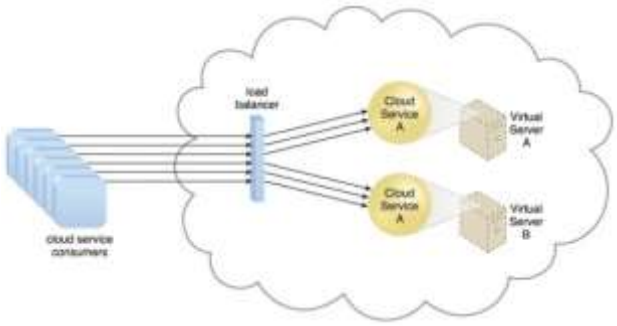
1. *Local Area Network*: Jenis jaringan area kecil tertentu adalah LAN (jaringan area lokal). Contohnya adalah jaringan komputer di ruang kelas, kampus, atau lingkungan bisnis. Teknologi Ethernet IEEE 802.3, dengan kecepatan transfer data 10 MB/s, 100 MB/s, dan 1 GB/s, sering digunakan dalam jaringan LAN. Teknologi nirkabel seperti Wi-Fi adalah pilihan lain untuk jaringan LAN selain menggunakan teknologi Ethernet.

2. *Metropolitan Area Network* : Jaringan area metropolitan, atau MAN, adalah jaringan yang berbasis di satu kota, memiliki kecepatan transmisi data yang tinggi, dan menghubungkan beberapa situs sambil tetap berada di kota yang sama. Jaringan MAN terdiri dari banyak jaringan LAN.

3. *Wide Area Network* (WAN): WAN adalah jaringan yang menghubungkan beberapa lokasi, seperti wilayah, kota, negara, atau benua.

2.4 Sistem Penyeimbang Beban (*Load balancing*)

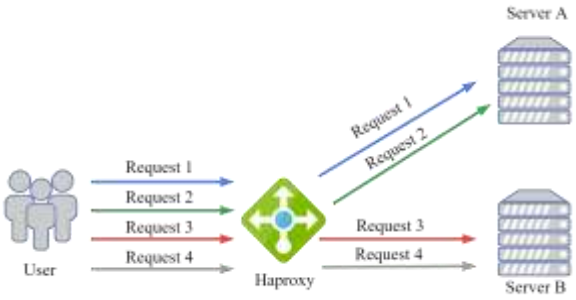
Load balancing adalah teknik jaringan komputer yang mendistribusikan lalu lintas beban kerja melalui banyak *server* atau *cluster server* untuk mencapai penggunaan sumber daya maksimum, meningkatkan *throughput*, mengurangi waktu respons, dan menghindari kelebihan beban. *Cluster server* adalah kumpulan perangkat *server* yang terhubung dan bekerja sama yang dapat dilihat sebagai satu sistem dalam banyak cara. *Cluster server* biasanya digunakan untuk meningkatkan kinerja dan ketersediaan pada satu *server* [9]. Penyeimbangan beban digunakan ketika satu *server* lokal diakses dengan lebih banyak pengguna atau permintaan layanan daripada yang dapat ditanganinya. Untuk mendapatkan penggunaan sumber daya terbaik, penyeimbangan beban mendistribusikan lalu lintas beban kerja secara merata di antara dua atau lebih *server*, jalur jaringan, CPU, *hard drive*, atau sumber daya lainnya. HaProxy adalah salah satu strategi *load-balancing*. Gambar 1 di bawah ini menunjukkan struktur *load balancing*.



Gambar. 1. Salinan Layanan *Cloud* A yang *overload* diimplementasikan pada *Server Virtual* B. Penyeimbang beban mengalihkan permintaan konsumen layanan *cloud* dan mengarahkannya ke *Server Virtual* A dan B untuk memastikan distribusi beban kerja yang merata [10].

2.5 HAproxy

HAProxy adalah proyek *open source* berlisensi GPLv2. Haproxy adalah penyeimbang beban TCP/IP dan *server* proxy yang dapat memuat-berbagi permintaan masuk dengan *server* multi-simpul. Akibatnya, beban *server* akan didistribusikan diantara *node server* yang tersedia. Ada beberapa opsi; ada yang sama, ada yang berdasarkan volume lalu lintas, dan ada yang unik. Setelah itu, kita akan melihat algoritma pembagian beban permintaan [11]. Gambar 2 dibawah ini menunjukkan konsep dari HAproxy.



Gambar. 2. Konsep HAproxy.

2.6 Round Robin

Round Robin DNS adalah dengan mengelola respons Sistem Nama Domain (DNS) untuk menjawab permintaan dari pengguna komputer sesuai dengan model statistik yang sesuai, penyeimbangan beban, atau penyediaan toleransi kesalahan dari beberapa layanan Protokol Internet redundan yang dihosting, misalnya, *server* Web, *server* FTP, dapat digunakan untuk mendistribusikan beban atau keseimbangan beban. DNS Round-robin beroperasi secara langsung dengan membalas permintaan DNS dengan daftar alamat IP dari berbagai *server* yang menghosting layanan yang sama, bukan hanya satu alamat IP. Menjalankan round robin didasarkan pada urutan alamat IP dari daftar yang dikembalikan. Urutan alamat IP dalam daftar fleksibel untuk setiap jawaban DNS [4].

Terlepas dari kapasitas atau permintaan server, metode *load balancing* round robin mendistribusikan permintaan layanan secara merata di antara semua *server*. Jika terdapat tiga *server* (A, B, dan C), permintaan layanan 1 akan dirutekan melalui load balancer ke *server* A, permintaan layanan 2 ke *serve*r B, permintaan layanan 3 ke *server* C, permintaan layanan 4 ke *server* A, dan permintaan layanan 5 ke *server* B. Jika semua *server* memiliki spesifikasi *server* yang sama, pendekatan ini dapat diterapkan. Algoritma ini didasarkan pada konsep *time-sharing*. Metode ini mirip dengan FCFS, hanya proaktif. Untuk membatasi waktu pemrosesan, setiap proses diberikan waktu CPU yang disebut waktu kuantum, yang biasanya antara 1 dan 100 milidetik. Proses dihentikan dan ditambahkan ke antrian siap setelah penghitung waktu berakhir. Algoritma round-robin adalah yang paling dasar dan digunakan oleh sebagian besar perangkat penyeimbang beban. Algoritma ini membagi beban secara progresif dan bergiliran dari satu *server* ke *server* berikutnya, membentuk *loop*.

2.7 Failover

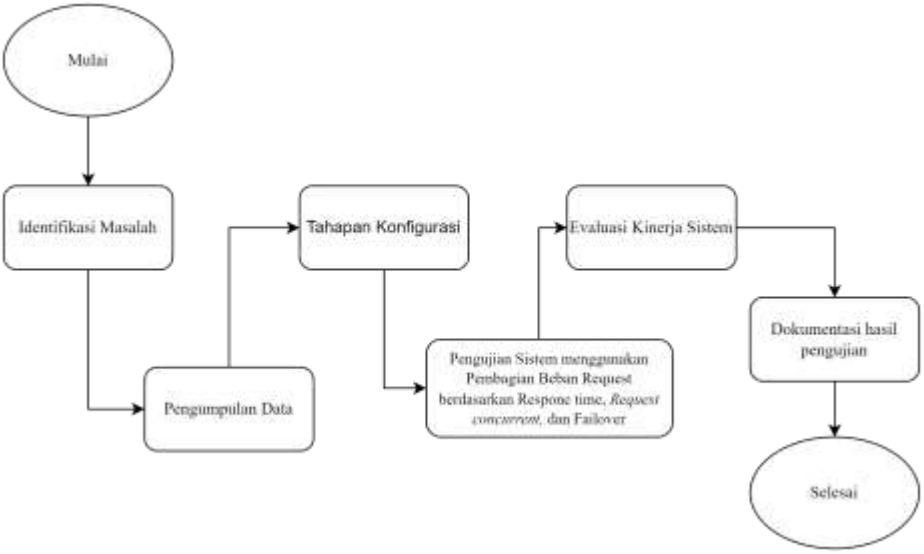
Failover merupakan sebuah teknik yang menggunakan beberapa sistem untuk mencapai sebuah *network* tujuan. Namun dalam keadaan normal hanya ada satu sistem yang digunakan. Sistem yang lain hanya digunakan apabila sistem utama terputus. *Failover* akan memindahkan sumber internet secara manual atau otomatis ke sistem cadangan apabila sistem utama mengalami kegagalan menggunakan metode *recursive gateway* [12].

2.8 Reverse Proxy

Reverse Proxy merupakan proxy yang mengarah internet dan digunakan sebagai *front-end* untuk mengontrol dan melindungi dari akses ke *private network*, sementara tugasnya meliputi penyamarataan beban, *caching*, deskripsi, dan otentikasi [13]. *Reverse Proxy* juga digunakan sebagai pengaman agar proses pertukaran *request* dari *user* ke *server* atau sebaliknya berjalan dengan aman. Tidak hanya itu, *Reverse Proxy* juga dapat melakukan pemampatan data. Data yang besar akan dimampatkan sehingga ukuran data menjadi lebih kecil. Hal ini dapat mempercepat proses pertukaran data. *Reverse Proxy* juga memiliki kemampuan untuk menyamaratakan beban trafik atau *load server* agar *server* tidak mengalami kegagalan (*down*).

3 Metodologi Penelitian

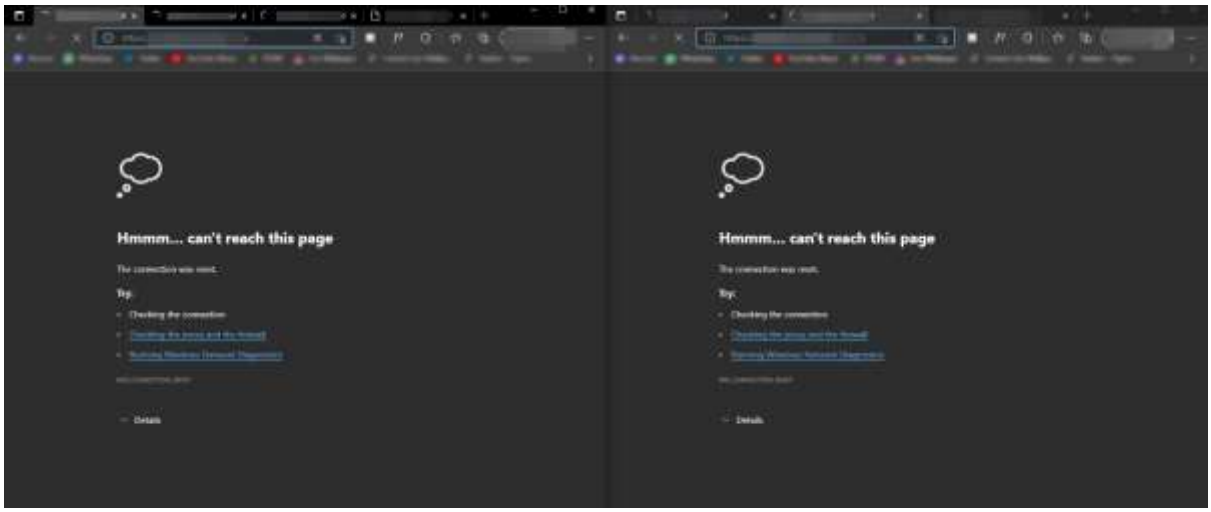
Dalam penelitian ini terdapat beberapa tahap yang akan dilakukan yaitu pemilihan metode *load balancing* digunakan untuk meningkatkan *server availability Learning Management System* (LMS) Universitas XYZ yang didukung dengan HAProxy dalam simulasi metode penelitian ini. Untuk tahap awal akan dilakukan Identifikasi Masalah. Proses selanjutnya yaitu Pengumpulan data, dan dilanjutkan dengan Tahapan Konfigurasi. Pengujian Sistem dilakukan dengan cara menggunakan Pembagian Beban *Request* berdasarkan *Response time, Request concurrent* dan *Failover* dari simulasi HAProxy terhadap *Learning Management System* (LMS) Universitas XYZ. Selanjutnya, menggunakan kriteria pengujian yang telah ditetapkan, Evaluasi Kinerja Sistem dilakukan pada penerapan pendekatan penyeimbangan beban dengan HAProxy. Nilai temuan uji parameter yang telah dilakukan dapat dilihat dari hasil evaluasi. Gambar 3 dibawah ini menunjukkan alur tahapan pada penelitian pembuatan sistem dengan *flowchart* yang digunakan untuk mencapai hasil yang dibutuhkan dalam penelitian ini.



Gambar. 3. Dalam studi kasus *Learning Management System* (LMS) Universitas XYZ, berikut alur penelitian yang akan diikuti untuk membuat sistem: *Cloud Load Balancing* Menggunakan HAProxy untuk Meningkatkan Server *Availability*.

3.1 Identifikasi Masalah

Pada bagian ini sedang diselesaikan untuk mencari masalah yang berkaitan dengan topik penelitian ini. Ruang lingkup masalah kemudian akan dikurangi untuk fokus pada sumbernya. Dalam penelitian ini terdapat permasalahan yang ditemukan pada server layanan Learning Management System (LMS) yang menggunakan moodle version 3.10.2 di Universitas XYZ namun untuk moodle verison yang digunakan tersebut tidak begitu berpengaruh terhadap server availability, tetapi yang ditemukan berpanguruh adalah masalah pada infrastruktur server availability yang menangani pengguna layanan Learning Management System (LMS) di Universitas XYZ yang menggunakan server tunggal yang mana jika terjadi kenaikan akses pengguna dan permintaan layanan daripada yang dapat ditangani dari server tunggal layanan Learning Management System (LMS) itu atau server tunggal tersebut mengalami kelumpuhan yang membuat layanan down tanpa ada backup layanan. Berikut gambar 4 dibawah adalah bukti server tunggal yang tersedia layanan LMS Universitas XYZ saat terjadi peningkatan aksess pengguna mengalami down.



Gambar. 4. *Server* tunggal yang tersedia layanan LMS Universitas XYZ saat terjadi peningkatan aksess pengguna mengalami down.

Dilakukan survei dengan mencoba langsung layanan *Learning Management System* (LMS) di Universitas XYZ, dan menemukan bahwa layanan *Learning Management System* (LMS) di Universitas XYZ hanya menggunakan infrastruktur *server* tunggal yang mana kejadian lumpuh *server* akan mengakibatkan *down*-nya layanan seperti

gambar 4.

3.2 Pengumpulan Data

Tahapan ini merupakan pendekatan yang bermanfaat untuk melakukan analisis data dan mengubahnya menjadi informasi, yang akan dilakukan untuk melihat permasalahan dalam kasus penelitian yang dibahas. Tahapan ini dapat dibagi menjadi dua bagian, sebagai berikut:

- Studi Literatur
  Langkah ini melibatkan penggabungan data dari beberapa sumber, seperti pencarian sumber, teori, konsep yang terkait dengan penelitian *Cloud Load Balancing* Dengan menggunakan HAProxy Dalam Meningkatkan *Server Availability* Pada Studi Kasus *Learning Management System* (LMS) Universitas XYZ yang akan dilakukan ditambah dengan sumber artikulasi, buku, jurnal, dan tesis, serta mengambil bahan dari internet untuk dijadikan landasan penelitian ini sebagai penguat solusi yang akan diterapkan untuk mengatasi permasalahan yang terjadi dalam penelitian ini.

- Studi Lapangan
  Pada bagian ini, data digabungkan dengan melakukan pengamatan di lapangan sebagai bagian dari proyek studi.

3.3 Tahapan Konfigurasi

Pada tahap ini dilakukan konfigurasi sistem *Server Cloud clustering load balancing* dengan HAProxy yang akan digunakan. Desain penelitian ini memiliki 7 tahapan alur di dalamnya yang dijelaskan pada Gambar 5 di bawah ini.

Gambar. 5. *Flowchart* Tahapan Konfigurasi Pada Sistem *Server Cloud Clustering Load Balancing* Dengan HAProxy.

3.4 Pengujian Sistem

Dilakukan dengan menggunakan 3 (tiga) metode yaitu menggunakan pembagian beban *request testing*, *Request concurrent* dan *Failover testing*. Pembagian beban *request testing*, metode pengujian sistem yang dilakukan dengan mengukur perbandingan antara *server* tunggal dengan *server cloud clustering load balancing* dengan HAproxy dalam meningkatkan kinerja *server* dalam melayani pengguna berdasarkan *response time*. *Request concurrent*, metode jumlah permintaan yang bisa dilayani dalam satu waktu seperti *Request Per Second. Failover testing*, metode yang dilakukan dengan melakukan simulasi pengujian fungsi *failover* ini bahwa dapat bekerja dengan baik dengan skenario akan mematikan *node* 1 dan *node* 2 atau mematikan salah satu *node*. Kemudian akan dicoba mengakses halaman simulasi *virtual sistem Learning Management System* (LMS) Universitas XYZ dengan alamat IP saat simulasi *down node* sedang berlangsung. Dengan mengukur tingkat keberhasilan *failover* dalam merespon *node* yang *down* dialihkan ke *node* yang normal dalam meningkatkan *server availability*. Fungsi *failover* sendiri berfungsi untuk mengalihkan permintaan akses jika ada 1 (satu) *server* atau lebih yang fungsinya pada protokol http/https mati atau *down*, permintaan ke *server* yang tidak berfungsi akan dialihkan ke *server* normal lainnya.

3.5 Evaluasi Kinerja Sistem

Pada tahap ini akan dilakukan evaluasi terhadap penerapan metode *cloud clustering load balancing* dengan HAProxy. diterapkan selama fase pengujian menggunakan parameter uji yang telah ditentukan. Hasil evaluasi akan menunjukkan nilai dari uji parameter yang dilakukan.

3.6 Dokumentasi Hasil Pengujian

Pada tahap ini, hasil pengujian dan evaluasi kinerja sistem didokumentasikan sebagai informasi yang dapat digunakan nantinya.

3.7 Kebutuhan Alat Hardware dan Software Dalam Penelitian

Ada dua jenis komponen yang digunakan dalam penelitian ini, yaitu perangkat keras (*hardware*) dan perangkat lunak (*software*), yang dapat dijelaskan sebagai berikut. Persyaratan perangkat keras dibagi menjadi 2 kategori, perangkat keras *client* dan perangkat keras *server*. Untuk perangkat keras *server* disesuaikan dengan simulasi penelitian ini yaitu menggunakan *server* virtual sesuai dengan spesifikasi yang ditemukan saat melakukan pengujian, sedangkan untuk *client* tidak memerlukan spesifikasi khusus. Untuk kebutuhan *software*, spesifikasinya terbagi menjadi 2 kategori, *software client* dan *software server*. *Client* hanya membutuhkan *web browser* untuk akses web dan *server* membutuhkan sistem operasi Ubuntu 18.04 untuk semua server, HAProxy untuk *load balancing server* dan Apache. Adapun spesifikasi teknis kebutuhan perangkat keras dan perangkat lunak ditunjukkan pada Tabel 1.

Tabel 1. Spesifikasi teknis kebutuhan perangkat keras dan perangkat lunak untuk implementasi

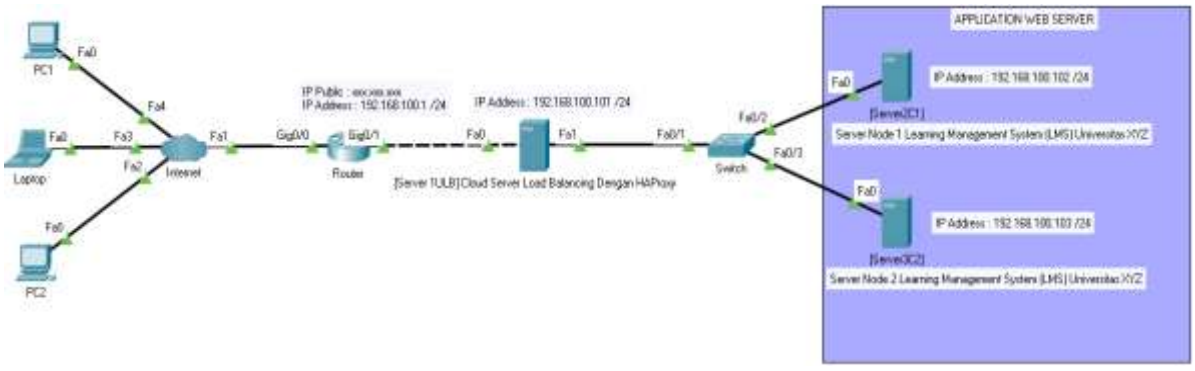
No.	Deskripsi	Spesifikasi
Kebutuhan *Client*		
Hardware		
1	Spesifikasi PC atau Laptop	Standart
Software		
1	Sistem Operasi	*All OS*

2	*Web Browser*	*All Browser*
3	*Stress Test Web Server*	*Apache Bench*
Kebutuhan *Server*		
Hardware		
1	*Router*	RouterBOARD 2011UiAS-2HnD
2	*switch*	RB941-2nD-TC (hAP-Lite2)
3	Server 1ULB *Cloud Server Load Balancing*	*Virtual Server*
4	Server 2C1 & 3C2 *Web Server*	*Virtual Server*
Software		
1	Sistem Operasi	Ubuntu 18.04
2	*Load Balancing*	HAProxy
3	*Web Server*	Apache2
4.	*Database*	MySQL Version 5.7
5.	*Learning Management System*	Moodle Version 3.10.2

4 Hasil dan Pembahasan

4.1 Arsitektur *Cloud Clustering Load Balancing* Dengan HAProxy

Perancangan *server cloud clustering* yang akan dibangun untuk *load balancing web server* dengan apache dan HAProxy adalah dengan menggunakan satu *server* sebagai *load balancing gateway server* yang sudah terpasang HAproxy, dua *server* sebagai *web server* utama. *Server gateway* berfungsi untuk mengatur pembagian beban *request* antara *web server* utama dan dapat seimbang dan kedua *server* tersebut terhubung dengan sistem *clustering web server*. Gambar 6 dibawah menggambarkan arsitektur *server cloud clustering load balancing* dengan HAProxy.



Gambar. 6. Arsitektur *server cloud clustering load balancing* dengan HAProxy.

Desain atau arsitektur *server cloud clustering load balancing* dengan HAProxy yang dirancang dapat dilihat pada Gambar 6 dengan spesifikasi konfigurasi sebagai berikut:

a. *Router*
- Gig 0/0 : xxx.xxx.xxx (IP Public dari ISP)
- Gig 0/1 : 192.168.100.1 (Ke server 1ULB *Cloud Server Load Balancing* Dengan HAproxy)

b. *Gateway Cloud Server Load Balancing* Dengan HAProxy [Server 1ULB]
- Eth [Fa0 & Fa1] : 192.168.100.101 (IP dari *Router* & IP ke *Clustering*)

c. Server Utama (2 *virtual server*)
- Server A [2C1] : 192.168.100.102
- Server B [3C2] : 192.168.100.103

4.2 Pengujian Pembagian Beban *Request Testing*

Pada tahap pengujian pembagian beban *request testing* yaitu dilakukan dengan mengukur perbandingan antara *server* tunggal dengan *server cloud clustering load balancing* dengan HAproxy dalam meningkatkan kinerja *server* dalam melayani pengguna berdasarkan *response time*, dan *request concurrent* . Berikut skenario pengujian yang akan dibahas dan tahapan pengujian yang akan dilakukan :

Dilakukan skenario simulasi *stress test server* dengan *tools* Apache Bench dari sebuah tes yang dibuat seperti command Parameter n adalah jumlah koneksi ke server tujuan, dengan contoh asumsi koneksi yang dibuat adalah 20000 koneksi untuk test 1, 30000 koneksi untuk test 2, dan 50000 koneksi untuk tes 3. Parameter c merupakan jumlah *request concurrent* (berbarengan) yang dibuat, dengan asumsi jumlah *request* yang dibuat adalah 10000 *request* dalam satu waktu untuk tes 1, 20000 request dalam satu waktu untuk tes 2, dan 20000 *request* dalam satu waktu untuk *test* 3. Parameter terakhir yaitu URL, halaman yang akan diproses oleh *web server* di-benchmark yaitu http://192.168.100.101 dengan pengulangan uji sebanyak 10 kali pada setiap test 1, 2, dan 3 dalam kondisi *server* tunggal serta *server cloud clustering load balancing* dengan HAproxy lalu diambil nilai rata rata *Request concurrent* dan *Response Time*. Untuk mengetahui pemerataan beban pada *server cloud clustering load balancing* dengan HAproxy serta dibuat kondisi salah satu *server cloud clustering load balancing* dengan HAproxy mati dari semua kondisi tes yang menjadi server tunggal, lalu hasil dari perbandingan antara *server cloud clustering load balancing* dengan HAproxy dengan *server* tunggal tersebut dievaluasi.

Setelah melakukan *stress test server* berhasil terhadap *web server cloud clustering load balancing* dengan HAproxy serta *server* tunggal dan testing sebanyak pengulangan 10 kali pada setiap test 1, 2, 3 lalu diambil value rata-rata maka pada tabel 2 berikut adalah tabel hasil dari testing yang telah dilakukan :

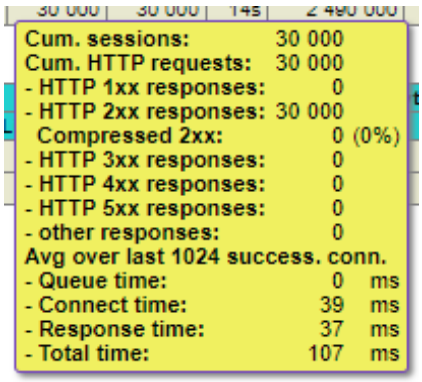
Tabel 2. Perbandingan antara *server* tunggal dengan *server cloud clustering load balancing* dengan HAproxy

Koneksi/*Request Concurrent*	*server* tunggal		*server cloud clustering load balancing* dengan HAproxy	
	Requests per second (#/sec [*mean*])	*Response Time* (ms)	*Requests per second* (#/sec [*mean*]	*Response Time* (ms)
20000/10000	463.14/sec	67 ms	524.31/sec	33 ms
30000/20000	515.81/sec	59 ms	593.34/sec	37 ms
50000/20000	502.86/sec	84 ms	593.33/sec	43 ms
Hasil Rata-Rata	493.93/sec	70ms	581.54/sec	37.6 ms

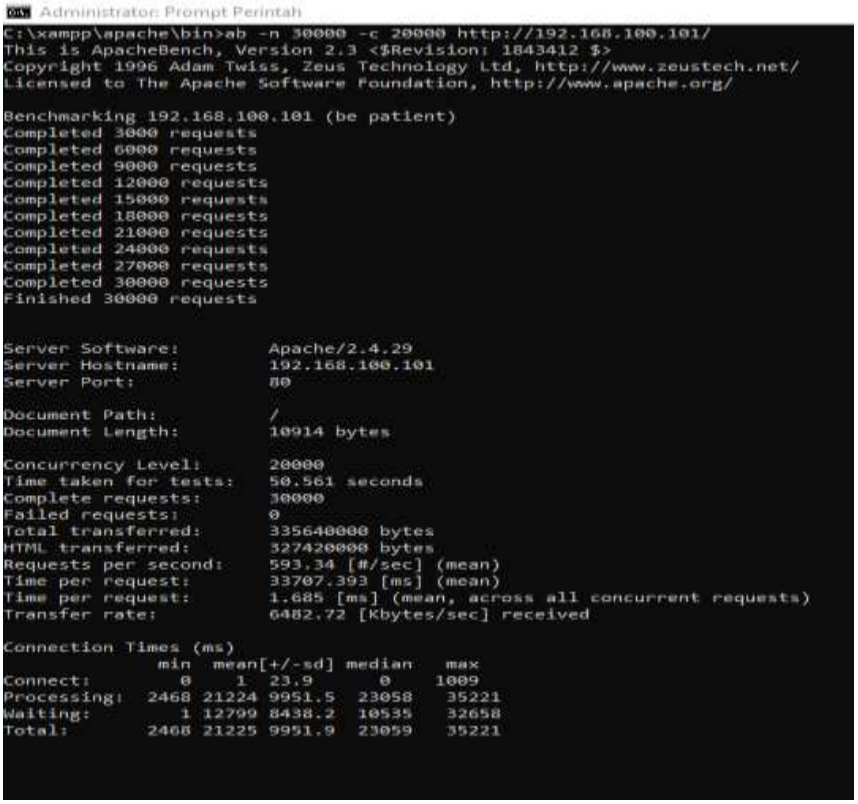
Gambar 6 memperlihatkan perbandingan secara grafik antara *server* tunggal dengan *server cloud clustering load balancing* dengan HAproxy, dengan server tunggal dan *server cloud clustering load balancing* dengan HAproxy dari nilai *Requests per second* (#/sec [*mean*]) dan *Response Time* (ms) yang telah diukur sebanyak pengulangan 10 kali pada setiap test 1, 2, 3 lalu diambil *value* rata-rata.

Gambar. 7. Grafik nilai *Requests per second* (#/sec [*mean*]) dan *Response Time* (ms).

Dari mengevaluasi pengujian server tunggal dan model server *cloud clustering load balancing* dengan HAproxy maka terlihat pada tabel 2 dan Gambar 7. Dari informasi yang didapat, implementasi *cloud clustering load balancing* dengan HAproxy memiliki nilai *response time* lebih kecil dari pada sebelum diterapkan *cloud clustering load balancing* dengan HAproxy dari rentang waktu pengujian berulang sebanyak 10 kali yang dilakukan lalu diambil nilai rata-rata. Pada uji koneksi dengan nilai permintaan sebanyak 20000 koneksi / 10000 *request concurrent* didapatkan nilai *response time* dari *cloud clustering load balancing* dengan HAproxy yaitu 33 ms, nilai ini lebih kecil bila dibandingkan dengan arsitektur tanpa *cloud clustering load balancing* dengan HAproxy, dengan nilai waktu respons 67 ms. Keadaan ini menunjukkan bahwa menggunakan cloud clustering load balancing menggunakan HAproxy meningkatkan waktu respons layanan dibandingkan dengan menggunakan server tunggal. Dalam hal kinerja, penggunaan cloud clustering load balancing dengan HAproxy dapat diterapkan tergantung pada nilai waktu respons yang baik dari layanan dan batas nilai request concurrent yang lebih tinggi untuk jumlah permintaan yang dapat diproses secara bersamaan. Implementasi cloud clustering load balancing dengan HAproxy pada sisi requests per detik yang dapat dilayani dan waktu respons menawarkan hasil yang sama pada pengujian koneksi dengan 30000/20000 request concurrent dan 50000/20000 request concurrent, seperti terlihat pada tabel 2. superior untuk penggunaan cloud clustering load balancing dari pada server tunggal.



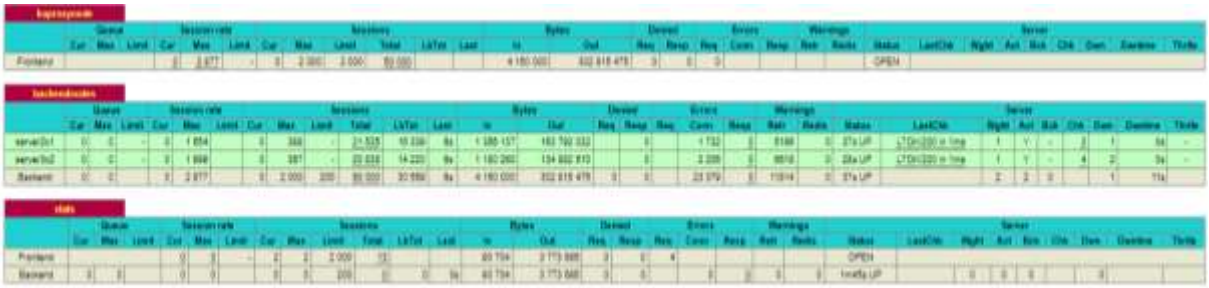
Gambar. 8. Hasil salah satu status proses pengujian *response time* terhadap *Server Cloud Clustering Load Balancing* dengan HAproxy.



Gambar. 9. Hasil salah satu status proses pengujian *Request Time Per Second* terhadap *Server Cloud Clustering Load Balancing* dengan HAproxy menggunakan Apache Bench.

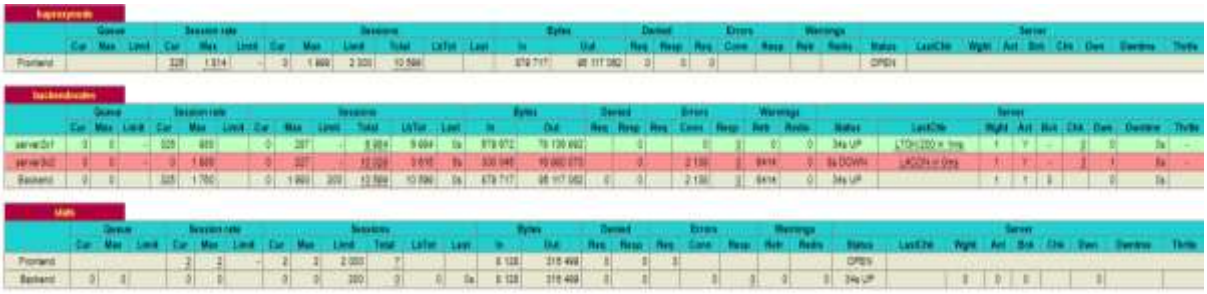
4.3 Pengujian *Failover Testing*

Pada tahap pengujian *failover testing* yaitu dilakukan dengan melakukan simulasi pengujian fungsi *failover* ini bahwa dapat bekerja dengan baik dengan skenario akan mematikan *node* 1 dan *node* 2 atau mematikan salah satu *node*. Kemudian akan dicoba mengakses halaman simulasi virtual sistem *Learning Management System* (LMS) Universitas XYZ dengan alamat IP saat simulasi *down node* sedang berlangsung. Dengan mengukur tingkat keberhasilan *failover* dalam merespon *node* yang *down* dialihkan ke *node* yang normal dalam meningkatkan *server availability*. Berikut gambar tahapan pengujian yang akan dilakukan dari skenario yang telah ditentukan :



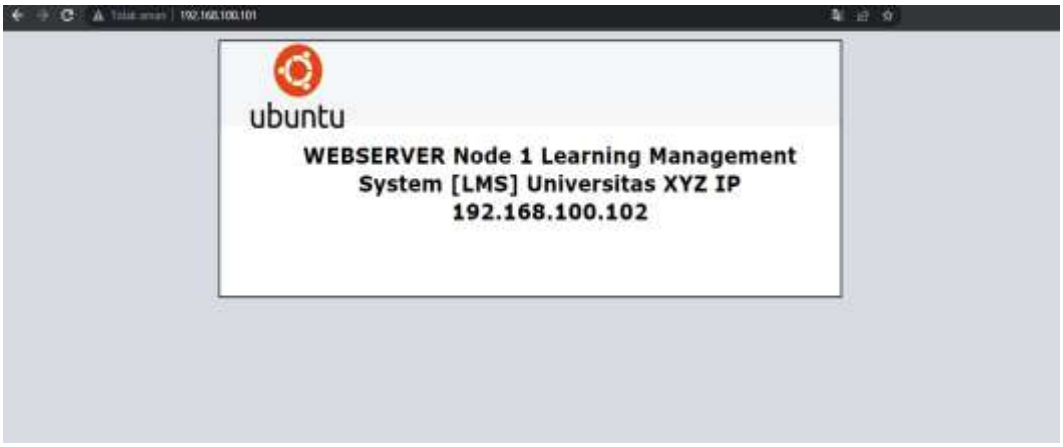
Gambar. 10. Tahapan *Cloud Clustering Load Balancing* dengan HAproxy normal tanpa ada yang *down.*

Gambar. 11. Tahapan *Cloud Clustering Load Balancing* dengan HAproxy mulai ada yang *down* dan mulai melakukan *failover* ke server yang normal.



Gambar. 12. Tahapan *Cloud Clustering Load Balancing* dengan HAproxy sudah *down* salah satu server serta layanan sudah di *failover* atau dilayani oleh server yang normal.

Pada gambar 12 hasilnya terlihat bahwa 1 server sudah dalam keadaan mati atau down ditandai dengan warna merah pada baris *node* 2 dengan IP 192.168.100.103. Kemudian pada gambar 13 dicoba akses halaman LoadBalancer http://192.168.100.101/ dengan *web browser* dan *server node* 1 "*WEB SERVER* Node 1 *Learning Management System* [LMS] Universitas XYZ IP 192.168.100.102" berhasil merespon yang menandakan *failover* pada HAproxy berjalan normal.



Gambar. 13. Halaman IP Address tersebut berhasil merespon dari *server node* 1 dan *server node* 2 sedang *down*.

Gambar. 14. Tahapan *Cloud Clustering Load Balancing* dengan HAproxy sudah mulai *up* kembali server yang *down* dan mulai *failover* layanan kedua server dengan *Load Balancing* serta *Response Time* 37 ms.

5 Kesimpulan

Ada kesimpulan yang dapat diambil dari hasil kedua pengujian yang telah dilakukan. Perbandingan rata-rata data menunjukkan bahwa *server* yang menggunakan *cloud clustering load balancing* dan HAproxy memiliki nilai *response time* 37,6 ms, lebih kecil dari *server* yang menggunakan *server* tunggal tanpa *load balancing*, yang memiliki nilai *response time* 70 ms. Ini menunjukkan bahwa server *cloud clustering load balancing* dengan HAproxy merespons lebih cepat dari pada *server* tunggal tanpa *load balancing*. *Requests per second* untuk *server*

cloud clustering load balancing dengan HAproxy adalah 581,54/dtk, lebih besar dari nilai *requests per second* untuk satu server tanpa *load balancing*, yaitu 493,93/dtk. Ini menunjukkan bahwa *cloud clustering load balancing* menggunakan HAproxy dapat menangani lebih banyak permintaan sekaligus lebih besar dari pada yang dapat dilakukan oleh server tunggal tanpa *load balancing*. Untuk hasil *failover* dari pengujian yang telah dilakukan menunjukan *failover* pada HAproxy berjalan normal karena pada gambar 12 Hasilnya menunjukkan bahwa 1 server sedang down, ditandai dengan tanda bintang merah pada node baris 2. Kemudian pada gambar 13 dicoba akses halaman http://192.168.100.101/ dengan *web browser* hasilnya *server node* 1 berhasil merespon *request* tersebut. Jadi, berdasarkan pengujian yang dilakukan dapat diketahui bahwa implementasi *cloud clustering load balancing* menggunakan HAproxy dalam meningkatkan ketersediaan *server* telah berhasil diterapkan. Dengan nilai *response time* yang lebih cepat daripada implementasi *server* tunggal tanpa *load balancing*, nilai *request concurrent* yang lebih tinggi, dan sistem *failover* yang berfungsi dengan baik.

Referensi

[1] Satria Harefa, H., Triyono, J., & Raharjo, S. (2021). IMPLEMENTASI LOAD BALANCING WEB SERVER UNTUK OPTIMALISASI KINERJA WEB SERVER DAN DATABASE SERVER. Jurnal JARKOM, 09(01), 10-20.

[2] Ardhian, D., Fatchur Rochim, A., & Didik Widianto, E. (2013). Analisis Perbandingan Unjuk Kerja Sistem Penyeimbang Beban Web Server dengan HAProxy dan Pound Links. Jurnal Teknologi Dan Sistem Komputer, 1(2), 28-33. https://doi.org/10.14710/jtsiskom.1.2.2013.28-33

[3] Dadi Riskiono, S., & Darwis, D. (2020). Peran Load Balancing Dalam Meningkatkan Kinerja Web Server Di Lingkungan Cloud. KREA-TIF : JURNAL TEKNIK INFORMATIKA, 8(2), 1-8. https://doi.org/10.32832/kreatif.v8i2.3503

[4] Budi Noviyanto, A., Kumalasari N, E., & Hamzah, A. (2015). PERANCANGAN DAN IMPLEMENTASI LOAD BALANCING REVERSE PROXY MENGGUNAKAN HAPROXY PADA APLIKASI WEB. Jurnal JARKOM, 3(2), 21-31.

[5] Salamah, I., Lindawati, M. Fadhli, & Kusumanto, R. (2020). EVALUASI PENGUKURAN WEBSITE LEARNING MANAGEMENT SYSTEM POLSRI DENGAN METODE WEBQUAL 4.0 . JURNAL DIGIT, 10(1), 1-10. https://doi.org/10.51920/jd.v10i1.151

[6] Sudiana, R. (2016). EFEKTIFITAS PENGGUNAAN LEARNING MANAGEMENT SYSTEM BERBASIS ONLINE. JPPM (Jurnal Penelitian dan Pembelajaran Matematika). JPPM, 9(2), 201-209.

[7] 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 , 03(02), 241-248. https://doi.org/10.25077/TEKNOSI.v3i2.2017.241-248

[8] Sumardi, S., & Asri Zaen, M. T. (2018). Perancangan Jaringan Komputer Berbasis Mikrotik Router OS Pada SMAN 4 Praya. JIRE (Jurnal Informatika & Rekayasa Elektronika), 1(1), 50-56. https://doi.org/10.36595/jire.v1i1.32

[9] Dadi Riskiono, S., & Pasha, D. (2020). Analisis Perbandingan Server Load Balancing dengan Haproxy & Nginx dalam Mendukung Kinerja Server E-Learning. Jurnal Telekomunikasi Dan Komputer, 10(2), 135-144. https://doi.org/10.22441/incomtech.v10i3.8751

[10] Erl, T., Puttini, R., & Mahmood, Z. (2013). Cloud computing: concepts, technology, & architecture. Pearson Education.

[11] Evianti, N., Mulyana Wihandar, A., & Kurniawan, A. (2021). AUTOMATION PROVISIONING DEV-OPS WEBSITE SERVER MENGGUNAKAN ANSIBLE DAN VAGRANT. JUNIF: Jurnal Nasional Informatika, 2(2), 72-91.

[12] Novianto, D., & Helmud, E. (2019). Implementasi Failover dengan Metode Recursive Gateway Berbasis Router Mikrotik Pada STMIK Atma Luhur Pangkalpinang. Jurnal Informatika Global, 10(1), 26-31. https://doi.org/10.36982/jig.v10i1.732

[13] Dwi Utomo, A., Rachmawati, Rr. Y., & Iswahyudi, C. (2017). ANALISIS DAN IMPLEMENTASI REVERSE PROXY SEBAGAI MEDIA KOMUNIKASI CLIENT SERVER MENGGUNAKAN APACHE (Studi Kasus Pada Lab. Jaringan Komputer IST AKPRIND YOGYAKARTA). Jurnal JARKOM, 05(2), 109-118.