Jurnal Masyarakat Informatika *ISSN: 2086-4930*
Vol. 13, No. 2, 2022 *e-ISSN: 2777-0648*

Infrastruktur High-Available Learning Management System Universitas Menggunakan Least-Connected Load Balancer

Lily Wulandari[1)], Imam Ramadhan[2)]

Pascasarjana Magister Teknologi dan Rekayasa[12], Universitas Gunadarma[12]
[1)] lily@staff.gunadarma.ac.id, [2)]i.rhadamant@gmail.com

Abstrak

IPB University berdiri sejak tahun 1963 sebagai salah satu perguruan tinggi ternama di Bogor. Pandemi membuat civitas akademik IPB University untuk melakukan kegiatan belajar mengajar jarak jauh. Namun, berjalannya Learning Management System (LMS) belum cukup untuk melayani ribuan pengguna tanpa kesalahan. High-availability menjadi desain lingkungan produksi esensial untuk meminimalisir kegagalan karena ada sistem lain yang dapat menggantikan sehingga proses produksi tidak terganggu. Untuk mewujudkan konsep high-availability pada LMS, diperlukan perencanaan skema sistem server yang matang. Penelitian ini mengimplementasikan konsep high-availability pada server LMS terdistribusi dengan menggunakan load balancing, web clustering, dan juga database clustering. Implementasi menggunakan serangkaian server basis data, server web, server NFS, server load balancing web, dan juga server load balancing basis data. Sistem terimplementasi telah diuji berdasarkan kebutuhan fungsional yang sesuai. Pengujian yang dilakukan ditinjau dari pengamatan terhadap waktu respon dan juga performa LMS. Hasil pengujian menunjukkan bahwa LMS telah mengaplikasikan konsep high-availability ketika terjadi suatu kesalahan pada salah satu server web. Pengujian juga menunjukkan bahwa kinerja server web telah seimbang menggunakan least-connected load balancer. Ketika salah satu server web tidak dapat diakses, maka server web yang lain mengambil request client sehingga sistem tetap dapat berjalan.

Kata kunci : *Database cluster, High-availability, Learning management system, Least-connected load balancing, Web cluster*

Abstract

IPB University has been stood since 1963 as one of the renowned colleges in Bogor. The global pandemic event made all its civitas to teach and learn remotely. Yet, depending on running the learning management system (LMS) solely is not enough to ensure serving thousands of client without any failures. High-availability becomes an essential production environment to minimalize failures as another systems replace so the production process remains undisturbed. In order to realize the high-availability concept to LMS, the university needs an intensive planning. This research implements the high-availability concept on distributed LMS servers utilizing server load balancing, web clustering, and database clustering. The implementation made use of database servers, web servers, NFS server, web load balancing server, and also database load balancing server. The implemented system had been tested referring to proper functional needs. It was reviewed on the LMS response time and system performance. Its result shows that the LMS had applied high-availability concept when fail occurs on any LMS web servers. The experiment also shown that web server performance has been balanced out using the least-

connected load balancer. When one web server could not be accessed, the other takes client requests so the system can keep running.

Keywords : *Database cluster, High-availability, Learning management system, Least-connected load balancing, Web cluster*

1 PENDAHULUAN

Tidak terlepas dari *event* global yang tengah terjadi di Indonesia, pandemi memaksa banyak pekerja untuk menghindari interaksi langsung dalam pekerjaannya. Terlebih, banyak juga dari mereka yang pada akhirnya kehilangan pekerjaan karena perusahaan ingin memangkas biaya pengeluaran seoptimal mungkin dalam keadaan ini. Bagi beberapa sektor, masih dimungkinkan bagi karyawan untuk melakukan pekerjaannya dari rumah yang kerap dikenal dengan istilah *Work from Home* (WFH).

IPB University merupakan salah satu kampus ternama di kota Bogor yang telah didirikan sejak tahun 1963 tidak luput dari pengaruh pandemi. IPB University dalam upaya bertahan dalam pandemi ini juga menerapkan sistem WFH bagi civitas akademiknya, baik dosen mau pun mahasiswa. Sistem ini kemudian mengusung kembali berbagai kebutuhan ekstra pada sektor teknologi informasi, secara spesifik *Learning Management System* (LMS). Yuni Fitriani menegaskan dalam penelitiannya bahwa berbagai LMS yang dikembangkan oleh pemerintah mau pun perguruan tinggi memiliki nilai manfaat lebih sebagai media pembelajaran daring selama pandemi berlangsung [1]. Pembelajaran daring seperti ini menuntut pergeseran pendidikan dari cara tradisional *teacher-centered* menjadi *student-centered*, seperti yang diutarakan Dean Ramadhan dalam penelitiannya [2].

IPB University menggunakan LMS berbasis MOODLE dalam pembelajaran *virtual* asinkronus maupun sinkronus. Berdasarkan penelitian yang dilakukan oleh Kristina Sara dkk., MOODLE dalam penerapannya agar mencegah penularan COVID-19 di kampus memiliki berbagai fitur yang menunjang sebagian besar kegiatan akademis mahasiswa [3]. Kegiatan belajar mengajar kini tidak menjadikan jarak sebagai keterbatasan. Dosen dapat tetap memberikan materi kuliah meski mahasiswa berada di berbagai tempat dengan menggunakan LMS. Pembelajaran daring, khususnya MOODLE dapat menumbuhkan ketertarikan dan antusiasme mahasiswa dalam belajar di tengah pandemi COVID-19 sebagaimana yang dipaparkan oleh Arlischa dan Arifin [4].

Meninjau seberapa pentingnya sistem pembelajaran berbasis web ini tentu server diharapkan untuk menolerir berbagai keadaan *failed*. Sayangnya, beberapa faktor baik internal mau pun eksternal mempengaruhi tujuan tersebut. Sebagaimana yang ditulis Ega Altania dan Sungkono dalam penelitiannya, server *down* menjadi salah satu kendala dalam pelaksanaan pembelajaran daring [5]. Kejadian yang cukup sering terjadi dan menyebabkan server *down* di IPB University di antaranya mati listrik dan gangguan pada jaringan *provider*, dua hal yang tentunya membuat performa maksimal server berkurang dalam melayani kegiatan belajar mengajar civitas kampus.

Konsep *high-availability* tepat diterapkan untuk mewujudkan skema server yang siap tersedia dan *reliable*. Upaya *failover* dapat dipenuhi ketika salah satu server berkendala dalam melayani permintaan dari pengguna. Teknologi komputasi terdistribusi adalah salah satu bentuk

upaya yang mengabstraksikan kemampuan *hardware* untuk mencapai *high-availability* dan sistem yang lebih andal untuk menangani kegagalan [6]. Server juga diharapkan dapat membagi beban yang diterima ke beberapa server lain di belakangnya sehingga beban *request* dari pengguna tidak terpusat di salah satu server saja, baik server web mau pun server basis data.

2 TINJAUAN PUSTAKA

Selama pandemi berlangsung, MOODLE menjadi salah satu alternatif LMS dengan segenap modul aktivitas yang dapat digunakan oleh pengajar. MOODLE mengatasi keterbatasan dalam belajar yang memerlukan interaksi langsung antara pengajar dan siswanya [7]. MOODLE menjadi aplikasi pembelajaran daring yang banyak digunakan di dunia. Aplikasi *open source* ini telah digunakan di 241 negara dengan 178000 situs terdaftar. Beberapa hal yang membuat MOODLE begitu populer di kalangan institusi pembelajaran adalah karena aplikasi tersebut merupakan aplikasi *open source*, mendapatkan dukungan komunitas global, dan juga memiliki konfigurasi yang fleksibel.

Yoji Yamato melakukan penelitian terkait virtualisasi dan pengelolaan server. Yamato juga memberikan proposal struktur server dengan membandingkan tiga tipe server, baik tradisional dan menggunakan teknologi virtualisasi. Ia membandingkan server *bare metal*, server hypervisor, dan juga server terkontenererisasi. Proposal yang ia kemukakan bergantung pada penggunaan dan harga. Server hypervisor tercatat memiliki nilai performa 60% bila dibandingkan dengan server *bare metal* [8]. Nilai ini cukup direkomendasikan dengan mempertimbangkan faktor yang disebutkan sebelumnya dan juga kebutuhan tempat yang cukup *compact* pada *data center*. Jyoti Shetty dan rekan menunjuk kontenerisasi dan virtualisasi hypervisor sebagai bahan pertimbangan dalam membuat infrastruktur server. Penelitian tersebut menyatakan bahwa dengan perbandingan server *bare metal*, server virtualisasi hypervisor, dan juga server terkontenerisasi, server yang menggunakan hypervisor unggul dalam hal *security* [9].

Server *load balancing* dapat menjadi kunci utama dalam konsep *high-availability*. Fahmi Apriliansyah dan rekan menyatakan bahwa server *load balancing* dapat menstabilkan dan menjaga keseimbangan web server, sehingga mampu melayani *traffic* tinggi dibandingkan dengan penggunaan *single web server* [10]. *Client* hanya akan melihat bahwa dirinya hanya mengakses satu server, namun di balik itu *load balancer* mengatur jalur trafik request client dengan berbagai algoritma yang dipilih sehingga akses dapat dirasakan dengan lancar. Hal ini dinyatakan dalam jurnal Arifwidodo dan rekan bahwa *load balancing* mendistribusikan dan mengalokasikan beban trafik di antara sumber daya yang tersedia, agar meningkatkan kinerja dan memaksimalkan *throughput* pada waktu respons minimum [11].

Sebuah penelitian lain yang dilakukan Setiawan dan rekan mengusung pembuatan *high-available web server* secara klaster. Implementasi tersebut mencakup beberapa perangkat lunak penunjang yang dapat dikorelasikan seperti Nginx untuk *web service* dan juga Maria DB untuk *database management service*. Setiawan menggunakan HAProxy dan Keepalive untuk mewujudkan konsep *high-availability* yang diangkat [12]. Di sisi lain, penelitian yang dilakukan Li dan rekan mengaplikasikan Nginx sebagai *load balancer*. *Load balancer* yang diaplikasikan pada server terklaster disimpulkan memiliki *response rate* yang lebih rendah, sehingga pengguna dapat mengakses web lebih cepat. Hasil evaluasi Li ditunjukkan pada Tabel

1 bahwa semakin banyak koneksi yang terhubung bersamaan, server yang diklaster cenderung memiliki *response time* yang stabil [13].

Tabel 1 *Response Time* antara *Single Server* dan Server *Cluster*

Jumlah koneksi terhubung bersamaan	*Single Server*	Server *Cluster*
50	6.3	4.1
100	6.7	4.1
150	7.4	4.6
200	44.7	5.2
250	105.3	5.2
300	268.9	5.7

Penelitian Akbar dan rekan mengombinasikan antara server *load balancing* dan juga *link load balancing* yang bertujuan mengurangi beban pada CPU server terdistribusi. Penelitian tersebut menggunakan algoritma *global first fit* sebagai metode *link load balancing* dan juga *least connection* sebagai metode *server load balancing*. Algoritma *least connection* mengatur beban server dengan parameter beban yang dimiliki server atau dengan metode *least loaded*. Semakin sedikit beban pada server, maka semakin besar prioritas server untuk dipilih. Dengan begitu, *request web client* diarahkan pada server yang memiliki total beban paling sedikit. Hasil evaluasi tersebut menyimpulkan bahwa penerapan metode kombinasi yaitu metode server dan *link load balancing* berhasil meningkatkan utilisasi jaringan dan juga mengurangi beban server [14].

Konsep *Network File System* (NFS) digunakan untuk mendistribusikan data di banyak server. NFS menjadi induk atau referensi dari server terdistribusi lain agar dapat menyamakan konten layanan. Sistem ini menjadi salah satu kunci dari layanan server *high-availability*, karena *request* dapat ditangani oleh banyak server [15].

Database clustering di samping itu merupakan teknik yang digunakan dalam mempertahankan kualitas data. Teknik ini menggunakan banyak basis data dalam operasinya dan mengelompokkan segenap basis data tersebut dalam satu lingkup yang sama. Apabila salah satu basis data gagal dalam melakukan pelayanan, maka basis data lain menggantikan basis data tersebut. Basis data pada *database clustering* selalu memiliki cadangan yang siaga. *Database clustering* dapat mencegah *single point of failure* dari sistem basis data [16].

Galera Cluster *Database Clustering* dapat diimplementasikan sebagai teknik untuk mewujudkan klaster basis data dengan metode *multi-master replication* untuk menjaga data antara basis data satu dan lainnya tetap sama. Galera Cluster menjamin perubahan data pada suatu node akan memicu perubahan pada node lainnya secara *realtime* [17]. Ketika terjadi kesalahan pada basis data, maka trafik akses data dapat dialihkan ke server basis data lain dalam klaster yang sama. Mengutip Mahendra Data dalam jurnalnya, *database clustering* menjadi lebih baik dengan konsep *multi-master*. Teknik ini menjadikan seluruh basis data yang ada pada klaster menjadi *master* tanpa ada *slave* secara definitif sama sekali. Mahendra juga menyimpulkan bahwa dengan berbagai jenis skenario kegagalan, jumlah node yang direkomendasikan dalam pembuatan *database clustering* adalah tiga, seperti yang ditampilkan pada Tabel 2.

Tabel 2 Hasil Percobaan Availabilitas dan Reliabilitas *Multi-master Database*

Jenis Kegagalan	2 Node	3 Node
Database service down	Normal	Normal
Connection down	Error	Normal
Server down	Error	Normal
10% packet loss	Normal	Normal

Database clustering dipadukan dengan Redis sebagai aplikasi *caching*. Zulfa memaparkan dalam penelitiannya bahwa aplikasi *caching* tersebut mempercepat akses data relasional [18]. Redis sebagai *cache* digunakan untuk meringankan kinerja dari server dengan cara menyimpan sementara konten di memori agar dapat digunakan kembali pada *request* yang sama. Aplikasi ini mendampingi basis data yang dipasang sehingga waktu respon dapat berkurang.

Read/write splitting merupakan istilah pemisahan transaksi *read* dan *write* pada basis data. Teknik ini merupakan salah satu implementasi dari bentuk *query routing* dengan mentransformasikan komunikasi basis data, membedakan transaksi *read* dan *write*, dan mengarahkan transaksi-transaksi tersebut sesuai dengan jalur yang telah dikonfigurasi. Umumnya pemisahan ini dilakukan pada lebih dari satu basis data. Basis data dikelompokkan ke dalam *reader* dan juga *writer*. Aplikasi-aplikasi yang mendukung *read/write splitting* seperti SQLProxy dan MySQL Proxy dapat digunakan sebagai lapisan penghubung yang bertindak sebagai pengelola *request* basis data. *Request* yang diterima dari *client* di dalam *connection pool* disaring dan diteruskan kepada server basis data sesuai dengan tujuannya. ProxySQL dapat digunakan sebagai *load balancer* dengan fitur *automatic failover*. Suatu transaksi basis data dapat dialihkan kepada basis data dengan *role* serupa.

3 METODE PENELITIAN

Penelitian ini menggunakan infrastruktur IPB University dalam studi kasus. Infrastruktur tersebut dikelola oleh direktorat khusus bernama Direktorat Sistem Informasi dan Transformasi Digital (DSITD). Skema arsitektur penelitian diimplementasikan pada server direktorat yang memiliki teknologi *Hyperconverged Infrastructure* di salah satu data center. Fitur utama pada teknologi tersebut adalah menjadi sebuah sistem perangkat yang memvirtualkan segala elemen sistem konvensional [19].

Pengembangan dalam penelitian ini menggunakan metode Network Development Life Cycle yang dimodifikasi. Alur yang dilakukan meliputi analisis kebutuhan, rancangbangun, implementasi, dan pengujian seperti yang diilustrasikan pada Gambar 1. Metode ini berupa siklus untuk mendapatkan hasil implementasi yang sesuai dengan kebutuhan.

Gambar 1 Alur Pengembangan

Secara fungsi, LMS yang dibuat diharapkan memenuhi kriteria sebagai berikut:

- Server *load balancer* mampu membagi beban kerja ke kedua server web LMS.
- Sistem tetap mampu beroperasi apabila salah satu server web LMS tidak aktif.

Mengacu pada kebutuhan fungsional sebelumnya, data yang dianalisa diperoleh dari dua pengujian pada skema infrastruktur. Pengujian pertama berfokus pada performa dari layanan *load balancer*. Pengujian dilakukan dengan menggunakan aplikasi Apache JMeter dan juga glances. Pengujian berupa simulasi pengguna untuk mengakses LMS secara bersamaan. Simulasi dieksekusi berdasarkan skenario banyak pengguna yang mengakses untuk mendapatkan rataan nilai *latency time*, *connected time*, dan persentase *successful request*. Pada saat yang sama, masing-masing server diukur penggunaan CPU (CPU total) menggunakan glances. Pengujian kedua sementara itu memiliki fokus kinerja availabilitas. Salah satu server web dinon-aktifkan untuk melihat keandalan sistem server ketika terjadi kesalahan pada salah satu server web LMS.

Selain itu, *latency* juga merupakan salah satu tolak ukur performa dari layanan server. *Latency* atau dikenal juga sebagai *delay* merupakan waktu jeda antara sebuah *request* dikirimkan dan *response* diterima kembali oleh pengguna. Kualitas *latency* dikategorikan pada Tabel 3 [20].

Tabel 3 Standar Kategori *Latency*

Kategori	*Latency* (ms)
Sangat baik	<150
Baik	150 s/d 300
Sedang	300 s/d 450
Jelek	> 450

4 HASIL DAN PEMBAHASAN

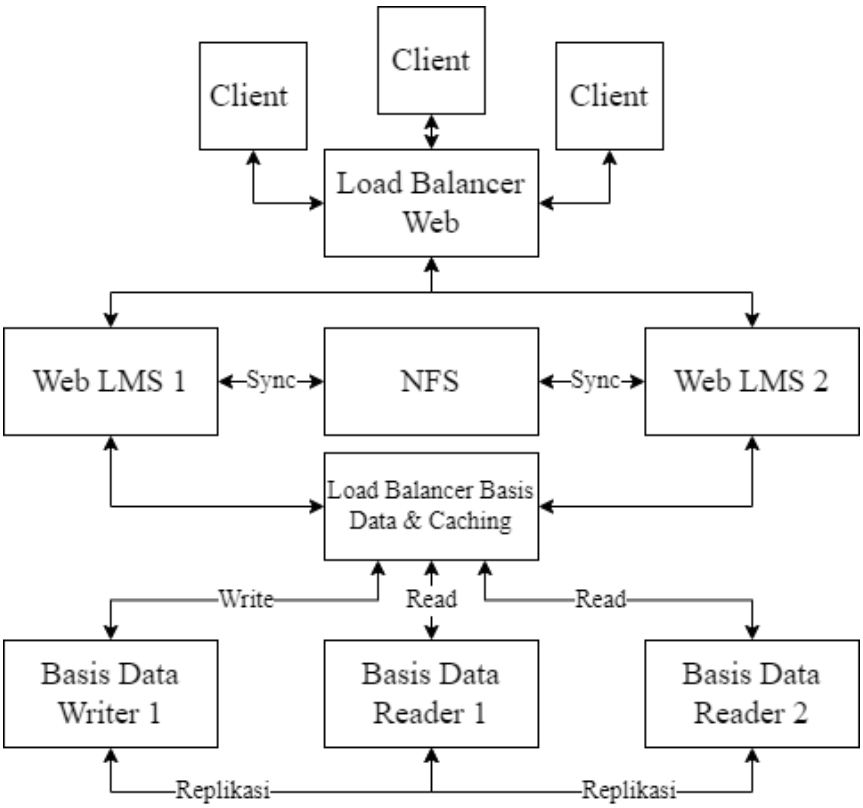
4.1 ANALISIS KEBUTUHAN

Infrastruktur yang dimiliki dan digunakan IPB University cukup bervariasi. Secara mayoritas, IPB Univeristy telah mengadaptasi teknologi virtualisasi dalam pengelolaan server di lingkungan produksi. Meski beberapa server masih berupa *bare metal*, server lain telah dikelola dengan teknologi hypervisor dan sebagian besar server menggunakan teknologi kontenerisasi. Infrastruktur pada penelitian ini menggunakan teknologi hypervisor tipe 1 yang memiliki akses langsung terhadap *hardware* yang digunakan. Terdapat delapan *virtual machine* yang bekerja di atas hypervisor tersebut, dengan peran masing-masing dalam membentuk LMS diteliti.

Penelitian yang dilakukan memiliki keterkaitan terhadap beberapa penelitian terdahulu. Basis data yang digunakan pada penelitian berjumlah tiga node. Mahendra Data dalam penelitiannya menyimpulkan bahwa tiga node basis data *multi-master* dapat menanggapi lebih banyak skenario kegagalan [10]. Jumlah node ini tentu menginspirasi penelitian agar mencakup konsep *high-available* pada basis data. Tidak hanya berfokus pada basis data, *load balancing* pada *web cluster* pun turut mewujudkan konsep *high-availability*. Penelitian yang dilakukan oleh Apriliansyah dan rekan dengan penggunaan berbagai algoritma. *Least-connection* dan *round robin* dalam penelitian mereka merupakan dua algoritma yang baik digunakan dalam

load balancing [4]. Algoritma *load balancing* diaplikasikan melalui *web service* Nginx terhadap klaster web yang menampung konten aplikasi MOODLE.

Pada Gambar 2, server yang digunakan berjumlah delapan dengan kategori *web server*, *database server*, *load balancer server*, *cache server*, dan *storage server*. Server basis data berjumlah tiga node, dengan komposisi satu server basis data sebagai *writer* dan dua lainnya sebagai *reader*. *Load balancer* basis data mengarahkan segala transaksi terhadap basis data menggunakan teknik *read/write split* pada aplikasi SQLProxy dalam klaster tersebut. Web LMS terdiri dari dua server yang tersinkronisasi kontennya melalui NFS. Semua *request client* terhadap web LMS dikelola oleh *load balancer* web. Implementasi semua server dilakukan secara terdistribusi untuk menghindari *single point of failure*.



Gambar 2 Arsitektur Sistem

4.2 RANCANG BANGUN

Implementasi delapan server yang digambarkan pada rancangan sistem pada Gambar 2 dilakukan dengan memasang dan melakukan konfigurasi aplikasi di masing-masing server. Aplikasi utama yang dipasang dapat dilihat pada Tabel 4. Sebagian besar server memiliki spesifikasi perangkat keras yang sama, seperti *virtual* CPU 1 *core*, 1GB RAM, dan *storage* 20GB. NFS memiliki spesifikasi *virtual* CPU 1 *core*, 1 GB RAM, dan *storage* 30GB. Kedua server web dan basis data *writer* memiliki spesifikasi *virtual* CPU 2 *core*, 2 GB RAM, dan *storage* 20GB.

Tabel 4 Spesifikasi Server

Server	*Virtual* CPU (*core*)	RAM (GB)	*Storage* (GB)	Aplikasi Utama
Load balancer Web	1	1	20	Nginx
Web Server 1	2	2	20	Nginx MOODLE
Web Server 2	2	2	20	Nginx MOODLE
NFS	1	1	30	NFS
Cache dan *Load balancer* Basis Data	1	1	20	Redis SQLProxy
Basis Data *Writer*	2	2	20	MariaDB
Basis Data *Reader* 1	1	1	20	MariaDB
Basis Data *Reader* 2	1	1	20	MariaDB

4.3 IMPLEMENTASI

Pada implementasi *database clustering*, basis data dibuat dalam satu klaster yang sama agar isi data saling tersinkronisasi. Hal ini dilakukan dalam upaya mencegah *single point of failure*, ketika salah satu basis data bermasalah. *Database clustering* pada penelitian diaplikasikan pada basis data yang dipasang MariaDB menggunakan teknik Galera *multi-master replication*. Penelitian menggunakan 3 server basis data yang dipasang aplikasi MariaDB sebagai 3 node di dalam klaster yang sama.

Klaster basis data tersebut dikelola oleh sebuah *load balancer*, dengan kemampuan *read/write splitting*. Setiap basis data dalam klaster tersebut dibedakan menjadi basis data *writer* dan juga basis data *reader*. SQLProxy menjadi aplikasi digunakan pada penelitian yang bertindak sebagai *load balancer* bagi basis data. Aplikasi tersebut menggunakan 2 *port* dalam pengelolaan klaster basis data, yaitu 6032 dan 6033. *Port* 6032 digunakan untuk kepentingan manajemen, sementara *port* 6033 merupakan *port* yang ditujukan user perwakilan untuk kepentingan operasional.

Server *load balancer* basis data juga dipasang *cache service* Redis. Redis adalah sistem basis data yang bersifat *key-value*. Tidak seperti *relational database* yang terdiri dari tabel-tabel yang memiliki relasi, sistem basis data *key-value* berbentuk pasangan *key* dan *value*.

Implementasi *web clustering* berkutat pada dua *web server* yang dijadikan sebuah klaster. Mirip dengan *database clustering*, konten pada kedua *web server* disinkronisasikan sehingga isi konten selalu sama. Hal ini diwujudkan dengan dukungan *storage server* yang dipasang NFS sebagai acuan kedua *web server* LMS untuk mengupdate isi konten webnya. Masing-masing server web diberikan perintah *mounting* agar dapat mengakses direktori induk yang dibuat.

Kedua server web juga dipasang Nginx sebagai aplikasi yang menyediakan layanan web. Hal ini berguna agar pengguna dapat mengakses LMS melalui *browser*. Nginx yang dipasang pada kedua server web murni digunakan untuk layanan web saja tanpa adanya server *load balancing*. Kedua server web tidak berhadapan langsung dengan pengguna sehingga server-server tersebut tidak dipasang pengamanan ekstra *Secure Socket Layer* (SSL) yang menjamin koneksi antara pengguna dan server. Aplikasi Nginx perlu diatur sehingga dapat memberikan layanan web sesuai dengan *domain* dan *root document* yang direncanakan.

4.4 PENGUJIAN

Para pengguna disimulasikan menggunakan aplikasi Apache JMeter. Masing-masing pengguna maya tersebut mewakili sejumlah dosen dan mahasiswa dalam melakukan akses LMS dari sekian banyak civitas IPB University. Pada penelitian ini, pengguna dipantau dalam usahanya melakukan akses terhadap halaman utama. Jumlah pengguna yang disimulasikan dikategorikan menjadi tiga skenario, yaitu 100, 500, dan 1000 orang berbeda. Satu per satu skenario diatur sedemikian rupa pada aplikasi JMeter sehingga LMS diakses secara bersamaan oleh sekelompok pengguna tersebut.

Pengujian yang dilakukan mengacu pada dua kebutuhan fungsional. Pengujian pertama dapat dilihat pada Tabel 5 terhadap rata-rata nilai *connected time* dan *latency time* yang didapatkan. *Latency* menjadi waktu yang dibutuhkan dalam *request client* terhadap server sampai menerima respon, sedangkan *connected time* merupakan waktu yang dibutuhkan client untuk menghubungi server. Pada ketiga skenario pengguna, tampak sistem server LMS yang diteliti dapat melayani *client* dengan cukup baik. Meski *latency time* yang ditunjukkan pada 1000 pengguna termasuk buruk, namun semua *request* dapat dilayani sistem server. Penambahan spesifikasi perangkat keras dapat mengurangi *latency time* pada jumlah pengguna tersebut.

Tabel 5 Hasil Pengujian Performa Menggunakan Apache JMeter

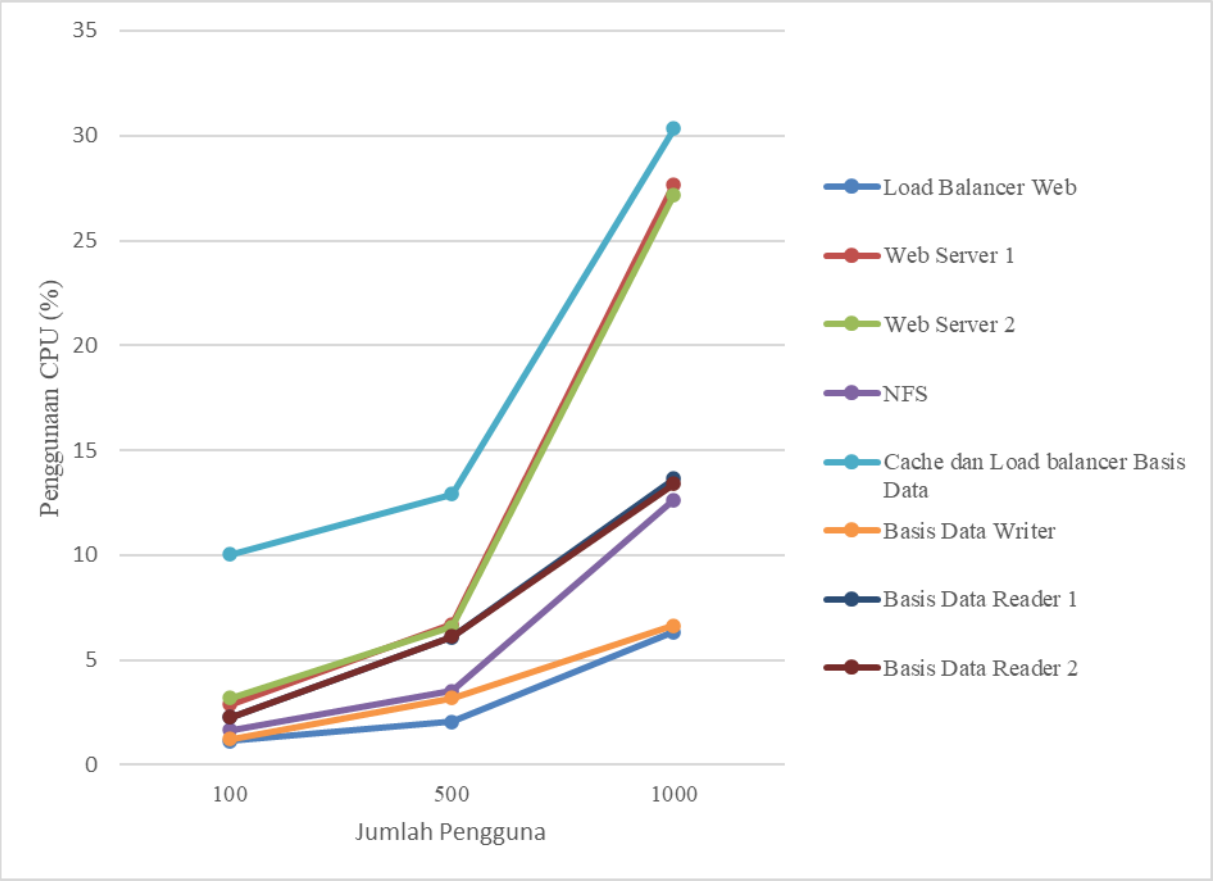
User	*Latency* (ms)	*Connected Time* (ms)	*Successful Request* (%)
100	55.761	5.440	100
500	64.254	4.955	100
1000	606.7	5.683	100

Pada pengujian yang sama, setiap server menjalankan aplikasi glances untuk mencatat rata-rata penggunaan CPU. Nilai total penggunaan CPU pada glances mewakili penggunaan CPU untuk ketiga skenario pengguna. Hasil pengujian dipaparkan pada Tabel 6 untuk setiap server. Berdasarkan rata-rata penggunaan CPU oleh 100, 500, dan 1000 pengguna dapat diamati bahwa server web LMS 1 dan server web LMS 2 memiliki jumlah yang hampir sama di angka maksimal 27%. Nilai yang tidak jauh berbeda didapatkan dari penggunaan *load balancer* web nginx dengan algoritma *Least-Connected*. *Load balancer* web membagi beban *request* yang diterima kepada server web di belakangnya berdasarkan beban kinerja paling sedikit. *Load balancer* pada *database clustering* juga terbaca bekerja dengan optimal. Transaksi yang diterima pada *load balancer* basis data menyaring permintaan dan meneruskan kepada basis data sesuai dengan *role* yang diberikan. Basis data *reader* 1 dan basis data *reader* 2 dijadikan sebagai contoh pada 1000 pengguna memiliki rataan CPU total yang juga mirip di angka 13%. Sementara itu, basis data *writer* pada jumlah pengguna yang sama tidak terlalu dibebani permintaan, karena sebagian besar transaksi basis data berupa *read*. Penggunaan CPU di masing-masing skenario bertambah mengikuti jumlah pengguna yang melakukan *request*, seperti yang diilustrasikan pada Gambar 3.

ISSN: 2086-4930
e-ISSN: 2777-0648

Tabel 6 Hasil Pengujian Performa Menggunakan Glances

Server	Rata-rata penggunaan CPU oleh 100 Pengguna (%)	Rata-rata penggunaan CPU oleh 500 Pengguna (%)	Rata-rata penggunaan CPU oleh 1000 Pengguna (%)
Load Balancer Web	1.15	2.077	6.35
Web Server 1	2.875	6.7	27.675
Web Server 2	3.2	6.589	27.2
NFS	1.675	3.533	12.65
Cache dan *Load balancer* Basis Data	10.025	12.922	30.325
Basis Data *Writer*	1.225	3.2	6.65
Basis Data *Reader* 1	2.275	6.089	13.65
Basis Data *Reader* 2	2.275	6.133	13.4



Gambar 3 Penggunaan CPU pada Setiap Server

Pengujian kinerja availabilitas dilakukan dalam menguji kebutuhan fungsional kedua. Tabel 7 adalah pengamatan yang dilakukan ketika salah satu atau bahkan kedua web server dinon-aktifkan. Tampak meski salah satu server non-aktif, LMS masih tetap dapat bekerja secara normal. Ketika server web LMS 1 non-aktif, pengguna masih dapat mengakses LMS. Sebaliknya, ketika server web LMS 2 non-aktif, pengguna masih dapat mengakses LMS secara normal.

Tabel 7 Uji Ketersediaan LMS

Simulasi Kesalahan Sistem	Kondisi
Server web LMS 1 non-aktif	LMS masih dapat diakses
Server web LMS 2 non-aktif	LMS masih dapat diakses
Server web LMS 1 dan 2 non-aktif	LMS tidak dapat diakses

5 KESIMPULAN

Berdasarkan pengujian yang telah dilakukan pada LMS, dapat disimpulkan bahwa sistem tersebut memiliki konsep *high-available*. Meski *latency time* terus meningkat, LMS masih dapat diakses dengan jumlah 100, 500, dan 1000 pengguna secara bersamaan. *Latency time* puncak yang didapatkan dari pengujian adalah 606.7 ms pada jumlah 1000 pengguna. *Request* yang diarahkan pada server web juga telah dibagi bebannya oleh *load balancer*, sehingga nilai penggunaan CPU antara server web LMS 1 dan LMS 2 tidak jauh berbeda di antara tiga skenario. Penggunaan CPU terbesar pada server LMS 1 dan LMS 2 tercatat pada skenario dengan jumlah 1000 pengguna. Poin-poin ini mengacu pada simulasi kesalahan sistem sehingga pengguna tetap dapat mengakses sistem LMS meski salah satu server web bermasalah.

DAFTAR PUSTAKA

[1] Y. Fitriani, "Analisa Pemanfaatan Learning Management System (LMS) Sebagai Media Pembelajaran Online Selama Pandemi COVID-19," *J. Inf. Syst. Informatics Comput.*, vol. 4, no. 2, pp. 1–8, 2020, doi: 10.52362/jisicom.v4i2.312.

[2] D. A. Ramadhan, F. Ardiansyah, J. Adisantoso, A. Asfarian, and Y. Nurhadryani, "Investigasi Awal Penggunaan Layanan Digital Perguruan Tinggi. Studi Kasus: IPB Mobile for Students," *J. Ilmu Komput. dan Agri-Informatika*, vol. 9, no. 1, pp. 37–46, 2022, doi: 10.29244/jika.9.1.37-46.

[3] K. Sara, F. L. Witi, and A. Mude, "Implementasi E-Learning Berbasis Moodle di Masa Pandemi Covid 19," *J. Adm. Educ. Manag.*, vol. 3, no. 2, pp. 181–189, 2020, doi: 10.31539/alignment.v3i2.1813.

[4] A. Ardinengtyas and A. N. Himawan, "Enhancing ELT Classroom Using Moodle E-Learning During the Pandemic: Students' and Teachers' Voices," *IJEE (Indonesian J. English Educ.*, vol. 1, no. 1, pp. 24–39, 2021, doi: 10.15408/ijee.v1i1.20220.

[5] E. Altania and Sungkono, "Pelaksanaan MOODLE di Masa Pandemi COVID-19 pada Mata Pelajaran Matematika Kelas 11 IPA," *J. EPISTEMA*, vol. 2, no. 1, pp. 83–88, 2021, doi: 10.21831/ep.v2i2.43251.

[6] J. G. A. Ginting, S. Ikhwan, and M. N. A. Ammar, "Analisis Performansi High Availability Web Server Pada Cluster GKE (Google Kubernetes Engine) Menggunakan Infrastruktur Google Cloud Platform," *InfoTekJar J. Nas. Inform. dan Teknol. Jar.*, vol. 5, no. 2, pp. 346–354, 2021, doi: 10.30743/infotekjar.v5i2.3577.

[7] L. Herayanti, M. Fuadunnazmi, and Habibi, "Pengembangan Media Pembelajaran Berbasis Moodle Pada Matakuliah Fisika Dasar Developing Moodle-Based Learning Media for Basic Physics," *Cakrawala Pendidik.*, no. 2, pp. 210–219, 2017, doi: 10.21831/cp.v36i2.13077.

[8] Y. Yamato, "Server Structure Proposal and Automatic Verification Technology on IAAS Cloud of Plural Type Servers," *IJIIS Int. J. Informatics Inf. Syst.*, vol. 1, no. 2, pp. 97–106, 2018, doi: 10.47738/ijiis.v1i2.104.

[9] J. Shetty, S. Upadhaya, H. S. Rajarajeshwari, G. Shobha, and J. Chandra, "An empirical performance evaluation of docker container, openstack virtual machine and bare metal server," *Indones. J. Electr. Eng. Comput. Sci.*, vol. 7, no. 1, pp. 205–213, 2017, doi: 10.11591/ijeecs.v7.i1.pp205-213.

[10] F. Apriliansyah, I. Fitri, and A. Iskandar, "Implementasi Load Balancing Pada Web Server Menggunakan Nginx," *J. Teknol. dan Manaj. Inform.*, vol. 6, no. 1, 2020, doi: 10.26905/jtmi.v6i1.3792.

[11] B. Arifwidodo, V. Metayasha, and S. Ikhwan, "Analisis Kinerja Load Balancing pada Server Web Menggunakan Algoritma Weighted Round Robin pada Proxmox VE," *J. Telekomun. dan Komput.*, vol. 11, no. 3, p. 210, 2021, doi: 10.22441/incomtech.v11i3.11775.

[12] R. Setiawan, D. P. Kartikasari, B. Rahayudi, F. Ilmu, K. Universitas, and P. Korespondensi, "Implementasi Arsitektur Web Server Cluster Menggunakan Single Board Computer," *Nasional*, vol. 8, no. 2, pp. 329–332, 2021, doi: 10.25126/jtiik.202184512.

[13] B. Li, J. Shang, M. Dong, and Y. He, "Research and Application of Server Cluster Load Balancing Technology," *Proc. 2020 IEEE 4th Inf. Technol. Networking, Electron. Autom. Control Conf. ITNEC 2020*, no. Itnec, pp. 2622–2625, 2020, doi: 10.1109/ITNEC48623.2020.9085082.

[14] H. P. Akbar, A. Basuki, and E. Setiawan, "Peningkatan Utilisasi Jaringan Distributed Storage System Menggunakan Kombinasi Server dan Link Load Balancing," *J. Teknol. Inf. dan Ilmu Komput.*, vol. 8, no. 3, p. 525, 2021, doi: 10.25126/jtiik.2021834294.

[15] F. W. Handono, H. Nurdin, and Sumarna, "Network File System Implementation on Computer-Based Exam," *Indones. J. Eng. Res.*, vol. 1, no. 1, pp. 20–26, 2020, doi: 10.11594/ijer.01.01.04.

[16] M. Data, G. Ramadhan, and K. Amron, "Analisis Availabilitas dan Reliabilitas Multi-Master Database Server Dengan State Snapshot Transfers (SST) Jenis Rsync Pada MariaDB Galera Cluster," *J. Teknol. Inf. dan Ilmu Komput.*, vol. 4, no. 1, p. 69, 2017, doi: 10.25126/jtiik.201741288.

[17] A. Heryanto and Y. Hartati, "Backup Database Dengan Multi Master Replikasi," *J. Ilm. Ilmu Komput.*, vol. 6, no. 1, pp. 7–12, 2020, doi: 10.35329/jiik.v6i1.118.

[18] M. I. Zulfa, A. Fadli, and A. W. Wardhana, "Application caching strategy based on in-memory using Redis server to accelerate relational data access," *J. Teknol. dan Sist. Komput.*, vol. 8, no. 2, pp. 157–163, 2020, doi: 10.14710/jtsiskom.8.2.2020.157-163.

[19] S. A. Azeem and S. K. Sharma, "Study of Converged Infrastructure & Hyper Converge Infrastructre As Future of Data Centre," *Int. J. Adv. Res. Comput. Sci.*, vol. 8, no. 5, pp. 2015–2018, 2017, doi: 10.26483/ijarcs.v8i5.3476.

[20] A. R. Rahmatika, S. Sukiswo, and E. D. Widianto, "Quality of Service Analysis of Long Term Evolution Network in Frequency Division Duplexing Mode in Semarang City," *Teknik*, vol. 41, no. 1, pp. 62–71, 2020, doi: 10.14710/teknik.v41i1.25850.