Jurnal Teknologi Informasi dan Ilmu Komputer (JTIIK) p-ISSN: 2355-7699
Vol. 12, No. 4, Agustus 2025, hlm. 759-770 e-ISSN: 2528-6579

IMPLEMENTASI LOAD BALANCING PADA GOOGLE CLOUD PLATFORM UNTUK MEMBANGUN ONLINE LEARNING

Endy Sjaiful Alim[1], M. Asep Rizkiawan[*2], Ahmad Subagyo[3]

[1,3]Universitas Muhammadiyah Jakarta, Tangerang Selatan, [2]Mekatronika, Politeknik Takumi, Bekasi
Email: [1]endy@uhamka.ac.id, [2]asep.mar@takumi.ac.id, [3]ahmad.subagyo@umj.ac.id
[*]Penulis Korespondensi

(Naskah masuk: 29 Agustus 2024, diterima untuk diterbitkan: 27 Agustus 2025)

Abstrak

Dalam era digital, institusi pendidikan menghadapi tantangan dalam menyediakan sistem pembelajaran daring yang andal, skalabel, dan responsif terhadap lonjakan pengguna. Salah satu permasalahan utama yang sering terjadi adalah *bottleneck* pada server web dan database, yang dapat menyebabkan penurunan performa saat jumlah pengguna meningkat secara signifikan. Penelitian ini bertujuan untuk mengatasi permasalahan tersebut dengan mengimplementasikan *load balancing* pada *Google Cloud Platform* (GCP) guna membangun platform pembelajaran daring berbasis Moodle yang optimal. Metode yang digunakan dalam penelitian ini mencakup perancangan dan implementasi infrastruktur berbasis layanan GCP, termasuk *Compute Engine* untuk *hosting* server web, *Cloud* SQL sebagai database terkelola, *Cloud Memorystore Redis* untuk *caching* guna mengurangi beban *query* pada database, serta *Cloud Filestore* untuk penyimpanan data. HTTPS *Load Balancer* digunakan untuk mendistribusikan lalu lintas pengguna secara merata ke beberapa *instance server*, sementara *autoscaler* diaktifkan untuk menyesuaikan kapasitas sumber daya secara dinamis sesuai kebutuhan pengguna. Hasil pengujian menunjukkan bahwa *bottleneck* utama pada sistem *e-learning* terjadi pada beban tinggi di database dan server web, yang dapat diatasi dengan *caching* dan *load balancing*. Implementasi ini memungkinkan sistem menangani lonjakan lalu lintas hingga 5.000 pengguna simultan. dengan penggunaan *moodle* data base mencapai 80,28 %, penggunaan *autoscaling* mencapai level 1,916. Utilisasi mulai menurun dan menunjukkan stabilisasi mendekati nilai 1. Stabilitas ini mengindikasikan bahwa *autoscaler* berhasil menyesuaikan jumlah *instance* dengan kebutuhan beban kerja, menjaga performa optimal aplikasi. Dengan demikian, penggunaan *load balancing* pada GCP terbukti meningkatkan keandalan, skalabilitas, dan efisiensi platform pembelajaran daring, serta memberikan panduan praktis bagi institusi pendidikan dalam mengadopsi teknologi *cloud* untuk mendukung kegiatan belajar-mengajar secara daring.

Kata kunci: *Load Balancing, Google Cloud Platform,* Pembelajaran Daring, *Moodle, Autoscaler,* Skalabilitas, Kinerja Sistem.

LOAD BALANCING IMPLEMENTATION ON GOOGLE CLOUD PLATFORM TO BUILD ONLINE LEARNING

Abstract

In the digital era, educational institutions face the challenge of providing a reliable, scalable and responsive online learning system to the surge of users. One of the main problems that often occurs is bottleneck on the web server and database, which can cause performance degradation when the number of users increases significantly. This research aims to overcome this problem by implementing load balancing on Google Cloud Platform (GCP) to build an optimal Moodle-based online learning platform. The method used in this research includes the design and implementation of GCP service-based infrastructure, including Compute Engine for web server hosting, Cloud SQL as a managed database, Cloud Memorystore Redis for caching to reduce query load on the database, and Cloud Filestore for data storage. HTTPS Load Balancer is used to distribute user traffic evenly across multiple server instances, while autoscaler is enabled to dynamically adjust resource capacity according to user needs. The test results show that the main bottleneck in the e-learning system occurs at high loads on the database and web server, which can be addressed by caching and load balancing. This implementation allows the system to handle traffic spikes of up to 5,000 simultaneous users. with moodle data base utilization reaching 80.28%, autoscaling utilization reaching a level of 1.916. This stability indicates that the autoscaler successfully adjusts the number of instances to the needs of the workload, maintaining optimal application performance. Thus, the use of load balancing on GCP is proven to improve the reliability, scalability, and efficiency of the online learning

platform, and provides practical guidance for educational institutions in adopting cloud technology to support online teaching and learning activities.

Keywords: *Load Balancing, Google Cloud Platform, Online Learning, Moodle, Autoscaler, Scalability, System Performance.*

1. PENDAHULUAN

Dalam dunia teknologi informormasi yang semua sudah mengarah pada digitalisasi, internet menjadi bagian utama dan terpenting dalam proses nya (Rizkiawan, Ramza, et al., 2023; Rizkiawan, Siregar, et al., 2023) termasuk dalam proses pembelajaran yang sudah mengadopsi metode pembelajaran secara *hybrid* (Mulyani et al., 2023). perkembangan teknologi informasi dan komunikasi telah mengubah cara pendidikan disampaikan, dari metode konvensional di ruang kelas menuju pembelajaran daring (*online learning*) (Ardiansyah & Awalludin, 2023). *Online learning* saat ini sudah menjadi bagian penting dari proses pembelajaran baik dari tingkat sekolah hingga perguruan tinggi, khusus nya pada tingkat perguruan tinggi menjadi bagian dari utama dalam proses pembelajaran, saat ini pembelajaran sudah mengadopsi pembelajaran jarak jauh dengan juga menggunakan metode *hybrid* (Alizadeh et al., 2019; Endy Sjaiful Alim, 2017; Kaur, 2020). Dalam era digital saat ini, pembelajaran jarak jauh dengan mengguanakan media *platform online* telah menjadi solusi utama bagi institusi pendidikan untuk menyediakan akses pendidikan yang fleksibel dan terjangkau (AGORMEDAH et al., 2020; Alim & Jin, 2017; Apoko & Sya'ban, 2022; Gumasing & Castro, 2023; Qazi et al., 2021; Sofi-Karim et al., 2023). Pandemi COVID-19 telah mempercepat adopsi pembelajaran *online*, Pembelajaran daring menawarkan fleksibilitas dan aksesibilitas yang tinggi, memungkinkan peserta didik untuk belajar dari mana saja dan kapan saja. Namun, keberhasilan pembelajaran daring sangat tergantung pada ketersediaan, performa, dan skalabilitas platform yang digunakan. Sebuah *Learning Management System* (LMS) seperti *Moodle* menjadi solusi populer di kalangan institusi pendidikan untuk mengelola konten pembelajaran, komunikasi, dan interaksi antara pengajar dan siswa. Namun, untuk menangani ribuan pengguna secara bersamaan, platform tersebut membutuhkan infrastruktur yang handal dan skalabel, menuntut infrastruktur yang mampu menangani lonjakan jumlah pengguna secara tiba-tiba dan memastikan pengalaman belajar yang lancar dan tidak terganggu (Dafa & Tewu, 2023; Nujid & Tholibon, 2023). Namun demikian ternyata terdapat Tantangan dalam Pembelajaran *online* seperti lonjakan *traffic*: pada saat sesi ujian, pendaftaran kursus, dan acara *online* besar lainnya dapat menyebabkan lonjakan *traffic* yang signifikan. Ketersediaan dan Keandalan: *Platform* harus selalu tersedia dan berfungsi dengan baik, bahkan selama puncak beban atau saat terjadi kegagalan server. Pengalaman Pengguna: Waktu respon yang lambat atau *downtime* dapat mengganggu proses belajar dan mengurangi kepuasan pengguna. Keamanan: Data pribadi siswa dan materi pendidikan harus dilindungi dari ancaman keamanan. Dengan meningkatnya jumlah pengguna yang mengakses *platform* pembelajaran *online*, masalah performa dan skalabilitas menjadi tantangan utama.

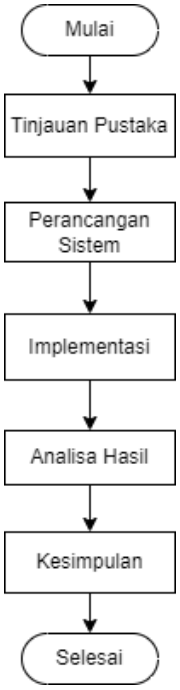
Google Cloud Platform (GCP) menyediakan serangkaian layanan yang dapat digunakan untuk membangun infrastruktur pembelajaran daring yang andal dan efisien. Salah satu komponen kritis dalam membangun infrastruktur tersebut adalah *load balancing. Load Balancing* adalah teknik yang digunakan untuk mendistribusikan beban kerja secara merata ke beberapa server, sehingga menghindari overloading pada satu server dan memastikan kinerja yang optimal (Boulmier et al., 2022; Mishra et al., 2020; Zhou et al., 2023). Selain itu, GCP menyediakan fitur *autoscaling* melalui *Managed Instance Groups* (MIGs) (Rout et al., 2023), yang memungkinkan sistem untuk secara otomatis menambah atau mengurangi jumlah *instance server* berdasarkan beban kerja aktual. Ini berarti bahwa saat ada lonjakan pengguna, sistem dapat menambah kapasitas secara dinamis, dan saat beban menurun, kapasitas dapat dikurangi untuk menghemat biaya. Kombinasi *load balancing* dan *autoscaling* memastikan bahwa *platform* pembelajaran daring dapat beroperasi dengan efisien dan responsif terhadap perubahan kebutuhan pengguna. Implementasi *Load Balancing* pada *Google Cloud Platform* (GCP) merupakan salah satu strategi penting untuk membangun *platform* pembelajaran *online* yang handal dan efisien. Universitas muhammadiyah Prof. DR. HAMKA sebagai salah satu universitas di jakarta yang memiliki mahasiswa cukup banyak harus adaptif dan peka terhadap kebutuhan dan tuntutan zaman yaitu dengan pembelajaran yang bisa dilakukan dimana saja dan memudahkan mahasiswa dalam mengakses pembelajaran. Menghadapi tantangan dalam menyediakan sistem pembelajaran daring yang handal dan mampu menangani lonjakan pengguna secara efisien. Untuk mengatasi permasalahan tersebut, Badan Pengembangan Teknologi Informasi (BPTI) UHAMKA mengembangkan platform *Moodle* berbasis *Google Cloud Platform* (GCP) dengan menerapkan *load balancing* dan *autoscaling* guna meningkatkan performa dan skalabilitas sistem pembelajaran daring.

Tujuan penelitian ini adalah membangun *platform Online learning* berbasis GCP, dengan memafaatkan *system load balancing* sehingga mampu melayani *user* dalam jumlah banyak secara *concurrent*. Adapun Batasan masalah dalam penelitian ini adalah, *Cloud Computing* di bangun di atas *platform* GCP. Sisi *security system* tidak masuk dalam pembahasan. Pembangunan *Online Learning* menggunakan aplikasi *moodle*. Pengalamatan menggunakan IP versi 4. Implementasi l*oad balancing* menggunakan *resources* yang ada di GCP. Parameter performansi pada *load balancing* akan di Analisa melalui tampilan *dashboard* yang tersedia pada GCP. Penelitian ini menggunakan pendekatan eksperimental dengan mengimplementasikan load balancing dan autoscaling pada GCP untuk mengoptimalkan performa LMS Moodle. Metode yang diterapkan mencakup, Perancangan Infrastruktur *Cloud* – Membangun sistem pembelajaran daring menggunakan layanan GCP seperti *Compute Engine (server hosting), Cloud* SQL (database), *Cloud Memorystore Redis (caching),* dan *Cloud Filestore* (penyimpanan data). Implementasi *Load Balancing* – Menerapkan HTTPS *Load Balancer* untuk mendistribusikan lalu lintas pengguna ke beberapa *instance* server guna menghindari overload pada satu server. Penerapan *Autoscaling* – Menggunakan *Managed Instance Groups* (MIGs) untuk menyesuaikan kapasitas server secara dinamis sesuai dengan beban kerja aktual. Pengujian dan Analisis Kinerja – Menganalisis performa sistem berdasarkan parameter seperti waktu respons, jumlah pengguna simultan yang dapat dilayani, serta efisiensi penggunaan sumber daya melalui *dashboard* pemantauan GCP.

Penelitian ini memberikan kontribusi dalam tiga aspek utama, pengembangan Infrastruktur LMS yang Optimal – Studi ini menunjukkan bagaimana kombinasi *load balancing* dan *autoscaling* dapat meningkatkan keandalan dan efisiensi sistem pembelajaran daring. Panduan Implementasi Teknologi *Cloud* dalam Pendidikan – Hasil penelitian ini dapat menjadi referensi bagi institusi pendidikan lain yang ingin mengadopsi *cloud computing* untuk mendukung pembelajaran daring. Evaluasi Kinerja LMS Berbasis Cloud – Penelitian ini menyajikan analisis performa yang dapat digunakan sebagai bahan pertimbangan dalam optimalisasi platform pembelajaran daring lainnya. Dengan demikian, penelitian ini diharapkan dapat memberikan solusi nyata bagi institusi pendidikan dalam mengelola sistem pembelajaran daring yang skalabel, efisien, dan andal.

2. METODE PENELITIAN

Bagian ini akan menguraikan proses penerapan mekanisme *load balancing* di *Google Cloud Platform* untuk membangun sistem pembelajaran *Moodle*. Metodologi penelitian ini mencakup beberapa tahapan, yaitu Tinjauan Pustaka, perancangan sistem, implementasi, dan analisis hasil. Berikut adalah rincian metodologi yang digunakan dalam penelitian ini. Rangkaian penelitian yang diterapkan sebagaimana terlihat pada gambar 1.



Gambar 1. Diagram Alur Metode Penelitian

Dalam melakukan penelitian ini, diperlukan banyak informasi detail. Penelitian ini menggunakan berbagai metode pengumpulan data. Yang pertama adalah Tinjauan Pustaka, di mana data dikumpulkan dengan mempelajari berbagai jenis bahan tulisan, seperti buku, jurnal, artikel, dan berbagai dokumen yang langsung terkait, Penelitian ini berfokus pada implementasi *load balancing* dalam lingkungan *cloud computing*, khususnya pada *Google Cloud Platform* (GCP). *Cloud computing* memungkinkan penyediaan sumber daya komputasi secara fleksibel dan efisien melalui internet (Taleb & Mohamed, 2020), sehingga banyak institusi pendidikan mengadopsi teknologi ini untuk mendukung pembelajaran daring. *Load balancing* adalah teknik distribusi lalu lintas ke beberapa server untuk menghindari *overload* pada satu server tertentu. Teknologi ini banyak digunakan dalam infrastruktur berbasis *cloud* untuk memastikan sistem tetap berjalan optimal meskipun terjadi lonjakan lalu lintas pengguna (Khan & Ali, 2023; Nancy et al., 2020). Perbandingan Metode *Load Balancing* di *Cloud* Untuk memahami keunggulan *load balancing* pada GCP, dilakukan perbandingan dengan beberapa metode *load balancing* yang digunakan pada *platform cloud* lainnya. Berikut adalah tabel perbandingan berbagai metode *load balancing*:

Tabel 1. Tabel perbandingan berbagai metode *load balancing*

Metode Load Balancing	Platform	Kelebihan	Kekurangan
HTTPS *Load Balancer*	*Google Cloud Platform* (GCP)	Skalabilitas tinggi, integrasi dengan autoscaling, performa optimal untuk aplikasi berbasis web	Konfigurasi awal cukup kompleks
Elastic Load Balancer (ELB)	*Amazon Web Services* (AWS)	Mendukung balancing untuk TCP, UDP, dan HTTP/S, serta kompatibilitas luas dengan layanan AWS	Biaya lebih tinggi dibandingkan dengan metode lain
Azure Load Balancer	*Microsoft Azure*	Mendukung balancing untuk trafik internal dan eksternal, serta integrasi dengan jaringan hybrid	Kurang optimal untuk aplikasi berbasis web dibandingkan solusi lainnya
Round Robin DNS	*Multi-platform*	Implementasi sederhana, tidak bergantung pada vendor cloud tertentu	Tidak memiliki fitur health check dan sulit menangani lonjakan lalu lintas
Least Connection Load Balancer	*Multi-platform*	Optimal untuk beban kerja yang tidak merata, karena menyeimbangkan berdasarkan jumlah koneksi aktif	Memerlukan konfigurasi tambahan dan monitoring yang lebih intensif

Dari perbandingan tersebut, HTTPS *Load Balancer* pada GCP dipilih dalam penelitian ini karena kemampuannya dalam menangani trafik tinggi, integrasi dengan autoscaling, serta performanya yang optimal untuk platform pembelajaran daring berbasis web seperti *Moodle*.

Pada langkah kedua ini, Perancangan sistem, Perancangan Arsitektur Sistem yang akan diimplementasikan pada *Google Cloud Platform*. Arsitektur ini mencakup: Pemilihan Komponen GCP: Pemilihan komponen-komponen yang diperlukan seperti *Compute Engine* untuk menjalankan *instance server*, *Cloud Storage* untuk penyimpanan data, dan *Cloud* SQL untuk manajemen basis data. Desain *Load Balancer*: Merancang *Load Balancer* yang akan digunakan, termasuk pemilihan jenis *Load Balancer* (HTTP(S) *Load Balancing* atau TCP/UDP *Load Balancing*) dan konfigurasi *routing* untuk mendistribusikan lalu lintas jaringan secara merata di antara *server*. Pengaturan *Auto-Scaling*: Menentukan konfigurasi *Auto-Scaling* yang akan digunakan untuk menyesuaikan kapasitas server secara dinamis berdasarkan jumlah permintaan dari pengguna. Tahap ke empat yaitu implementasi: Pada tahap ini, implementasi dilakukan untuk menerapkan perancangan sistem yang telah dibuat pada tahapan sebelumnya, Pada tahap ini, arsitektur yang telah dirancang akan diimplementasikan di lingkungan *Google Cloud Platform*. Implementasi ini mencakup: Penyebaran (*Deployment*) Aplikasi Pembelajaran *Online*: Aplikasi pembelajaran online yang telah dikembangkan akan di-*deploy* ke *instance server* yang berjalan di *Compute Engine*. Konfigurasi *Load Balancer: Load Balancer* yang telah dirancang akan dikonfigurasi untuk mendistribusikan lalu lintas ke beberapa *instance server* sesuai dengan arsitektur yang telah dibuat. Pengaturan Keamanan dan Monitoring: Penerapan pengaturan keamanan seperti *Firewall*, serta konfigurasi monitoring menggunakan *Google Cloud Monitoring* untuk memantau performa sistem. Tahap selanjutnya adalah melakukan analisa terhadap hasil yang telah ada seusai dengan peranacngan sistem yang telah diimplementasikan. Setelah mendapatkan hasil kemudian menarik kesimpulan dari penelitian yang dilakukan.

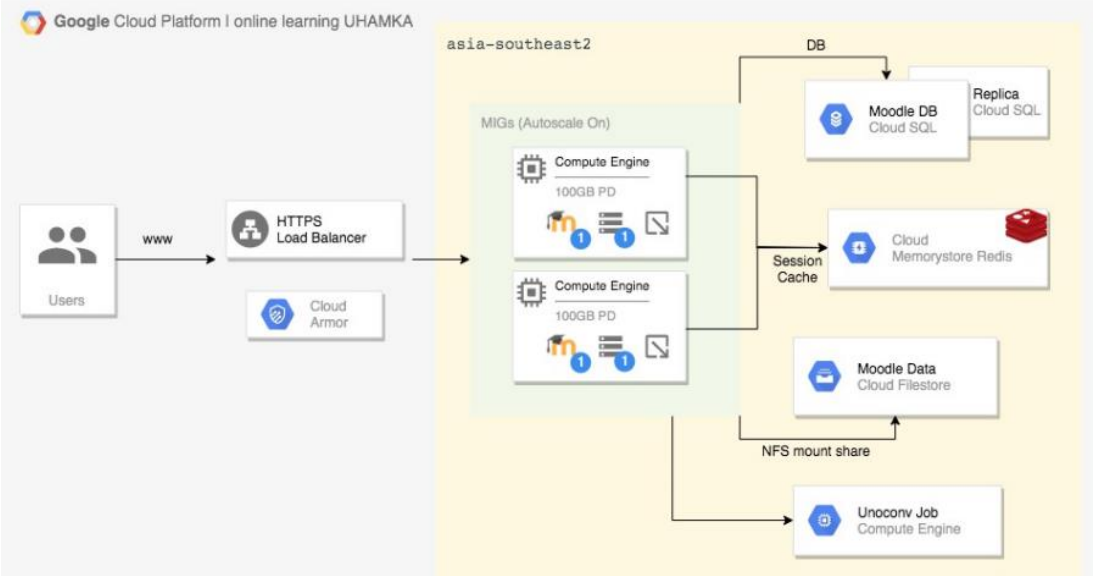
3. HASIL DAN PEMBAHASAN

3.1. Perancangan sistem Arsitektur *Online Learning* UHAMKA

Pada gambar 1. menunjukkan arsitektur infrastruktur untuk *platform* pembelajaran *online* UHAMKA yang diimplementasikan menggunakan *Google Cloud Platform* (GCP). Diagram ini menggambarkan berbagai komponen yang bekerja bersama untuk mendukung aplikasi *Moodle* dan memastikan kinerja, keandalan, serta skalabilitas yang optimal.

Pada arsitektur di atas beberapa *component* atau *product* dari *Google Cloud Platform* (GCP) yang digunakan untuk implementasi *e-learning* pada UHAMKA adalah sebagai berikut:

1. *Users*, Ini adalah pengguna akhir yang mengakses *platform* pembelajaran *online*. Pengguna mengirimkan permintaan melalui *browser web* atau aplikasi ke *server Moodle*.
2. GCP *Load Balancer*: adalah *product* dari layanan GCP yang dikhusus untuk pembuatan layanan *Load Balancer* (LB) *secara cloud*. Ada beberapa macam konfigurasi LB pada GCP yaitu *Load Balancing* HTTPS, *Load Balancing* TCP, *Load Balancing* UDP, namun pada implementasi kali ini menggunakan HTTPS. *Load balancer* akan bertugas sebagai operator untuk mendistribusikan trafik ke *Virtual Machine* (VM).
3. *Cloud Armor*, Fungsi *Cloud Armor* berfungsi sebagai pelindung aplikasi dari ancaman keamanan seperti serangan DDoS. Ini bekerja dengan menerapkan kebijakan keamanan yang membatasi akses berdasarkan aturan tertentu (misalnya, IP *filtering*, geolokasi, dll.), memastikan bahwa hanya lalu lintas yang sah yang bisa mencapai aplikasi.



Gambar 2. Arsitektur *Online Learning* UHAMKA

4. MIGs (*Managed instance groups*): adalah *product* dari layanan GCP yang salah satu fungsinya digunakan untuk penskala *service* secara otomatis, *service* dalam *use case* kali ini adalah aplikasi *Moodle*. MIGs akan berperan untuk menambah atau mengurangi VM berdasarkan beban yang sudah diatur sebelumnya.
5. *Compute Engine*: adalah *product* dari GCP yang digunakan untuk membuat *Virtual Machine*. Fungsi *Instance Compute Engine* ini adalah *server* utama yang menjalankan aplikasi *Moodle*. *Moodle* adalah aplikasi *Learning Management System* (LMS) yang digunakan untuk mengelola dan menyampaikan konten pembelajaran secara *online*. *Auto-Scaling: Instance* ini dikelola oleh *Managed Instance Groups* (MIGs) dengan fitur *Auto-Scaling* aktif, yang berarti sistem dapat menambah atau mengurangi jumlah instance secara otomatis berdasarkan beban kerja, menjaga kinerja aplikasi tetap optimal saat jumlah pengguna naik atau turun. *Persistent Disk*: Setiap *instance* menggunakan persistent disk 100 GB untuk penyimpanan data lokal yang diperlukan oleh aplikasi.
6. *Cloud SQL*: adalah *product* dari GCP yang dapat digunakan untuk membuat mesin SQL seperti *MySQL*, *PostgreSQL*, dan *SQL Server*. Dalam implementasi ini, *Cloud SQL* digunakan untuk menyimpan *database* aplikasi *Moodle*. Replikasi *Database*: Diagram menunjukkan adanya replikasi *database*, yang berarti data secara otomatis direplikasi ke *instance database* lain untuk meningkatkan keandalan dan ketersediaan data. Ini membantu menjaga data tetap tersedia bahkan jika satu *instance database* mengalami kegagalan.
7. *Cloud Memorystore Redis*: adalah *product* dari GCP yang dapat digunakan untuk membuat mesin *NoSQL Redis*. Dalam implementasi ini, Redis digunakan sebagai MUC (*Moodle Universal Cache*) dan *session* untuk menghandle *cache* dan *session* pada semua VM yang berjalan agar data tetap konsisten meskipun user mengakses VM yang berbeda. *Session Cache*: Redis berfungsi sebagai *session cache*, menyimpan informasi sesi pengguna agar akses dan interaksi menjadi lebih cepat.
8. *Cloud Filestore*: adalah *product NFS* dari GCP. Dalam implementasi kali ini, *Filestore* digunakan untuk menyimpan *User* Data (*moodledata*) agar semua data yang masuk dari VM yang berbeda tercentral di satu tempat dan tetap konsisten meskipun di akses melalui VM yang berbeda. NFS *Mount Share*: *Instance Compute Engine* mengakses *Filestore* melalui NFS mount, memungkinkan berbagai instance berbagi data yang sama secara efisien.
9. *Unoconv Job* : adalah *Compute Engine* yang bertugas untuk menghandle pemrosesan *data document* pada masing - masing *Compute Engine* yang menjalankan aplikasi *Moodle*, Mesin *Unoconv job* tidak perlu dibackup ketika proses migrasi, karena mesin ini hanya untuk melayani pemrosesan dokumen dan hasilnya akan di simpan pada *moodle* data yang ada pada *Cloud Filestore*. *Unoconv* (*Universal Office Converter*) adalah layanan yang digunakan untuk mengkonversi dokumen ke format lain (misalnya, dari *Word* ke PDF).

Perbandingan dengan Arsitektur *Cloud* Umum

Untuk memperjelas keunggulan rancangan sistem ini, berikut adalah perbandingan dengan arsitektur *cloud* tradisional (umum) atau *On-Premise / Cloud Non-Optimal*:

Tabel 2. Perbandingan dengan Arsitektur *Cloud* Umum

Fitur	**On-Premise / Cloud Non-Optimal**	**Arsitektur yang Diusulkan (GCP)**
Hosting LMS	Server fisik atau satu VM statis	Beberapa *instance Compute Engine dalam Managed Instance Groups*
Load Balancing	Manual atau tidak ada	HTTPS *Load Balancer* otomatis
Autoscaling	Tidak tersedia atau manual	*Autoscaler* berbasis beban kerja
Database	Database lokal atau *self-managed*	*Cloud SQL (managed database)*
Caching	Tidak ada *caching*	*Cloud Memorystore Redis*
Penyimpanan Berbagi	*File system* lokal atau NFS	*Cloud Filestore*
Monitoring & *Logging*	Manual atau tidak *real-time*	*Cloud Monitoring & Logging*

Perbedaan Utama, Sistem *On-Premise / Cloud Non-Optimal*: Biasanya menggunakan satu atau beberapa VM dengan konfigurasi manual tanpa *autoscaling* dan *load balancing* otomatis. *Database* dan penyimpanan mungkin masih berbasis lokal atau di-*host* tanpa pengelolaan otomatis. Sistem *Cloud* Optimal (GCP): Menggunakan layanan *cloud-native* seperti *managed databases*, *autoscaling, load balancing*, dan *caching* untuk meningkatkan performa dan efisiensi. Keunggulan utama dari rancangan ini dibandingkan dengan arsitektur *cloud* tradisional adalah: Skalabilitas Otomatis: Dengan *Managed Instance Groups* dan *Autoscaler*, sistem dapat menangani lonjakan pengguna secara dinamis. Kinerja Optimal: *Load Balancer* memastikan distribusi beban kerja yang merata untuk menghindari *overload* pada satu server. Efisiensi Sumber Daya: *Cloud SQL* dan *Cloud Memorystore* mengurangi beban server, mempercepat waktu akses data. Keandalan Tinggi: Dengan adanya pemantauan otomatis, sistem lebih cepat merespons gangguan atau kegagalan.

3.2. Detail Konfigurasi

Mekanisme Kerja *Load Balancer*, *Load balancing* adalah teknik yang digunakan untuk mendistribusikan lalu lintas jaringan ke beberapa server *backend* secara merata. Dalam penelitian ini, digunakan HTTPS Load Balancer pada *Google Cloud Platform* (GCP) untuk menangani permintaan pengguna secara efisien. Mekanisme kerja *load balancer* pada sistem yang diusulkan adalah sebagai berikut: Pengguna mengakses *platform Moodle* melalui URL yang diarahkan ke *Load Balancer*. HTTPS *Load Balancer* menerima permintaan dan menentukan *instance backend* yang paling optimal berdasarkan strategi distribusi beban yang dipilih. Jika suatu *instance server* mengalami *overload* atau gagal, *health check* akan mendeteksi dan mengalihkan lalu lintas ke *instance* lain yang sehat. Jika beban lalu lintas meningkat, *autoscaler* akan menambah jumlah *instance server* untuk menangani lonjakan pengguna.

Tabel 3. Pengaturan *Load Balancer* konfigurasi yang digunakan dalam *load balancer*

Parameter	**Nilai Konfigurasi**
Jenis Load Balancer	HTTPS Load Balancer
Backend Service	Managed Instance Groups (MIGs)
Protocol	HTTP(S)
Session Affinity	None (agar distribusi lebih merata)
Health Check	TCP Port 80 dengan threshold 3x retry
Caching	*Cloud CDN diaktifkan*
Timeout Connection	30 detik

Mekanisme Kerja *Autoscaler*, *Autoscaler* bertugas menyesuaikan jumlah *instance server* berdasarkan beban kerja yang terjadi. Dalam implementasi ini, *autoscaler* bekerja dengan mekanisme berikut: Memonitor pemakaian CPU dan jumlah permintaan per detik pada setiap *instance.* Jika pemakaian CPU melebihi *threshold* (misalnya 80%), *autoscaler* akan menambahkan *instance* baru untuk menangani beban tambahan. Jika beban menurun, *autoscaler* akan mengurangi jumlah *instance* untuk menghemat sumber daya dan biaya operasional. *Instance* baru yang ditambahkan secara otomatis akan terhubung ke *Load Balancer* untuk memastikan distribusi beban tetap merata.

Tabel 4. Pengaturan Pengaturan *Autoscaler* konfigurasi yang digunakan dalam untuk *Autoscaler*.

Parameter	**Nilai Konfigurasi**
Jenis Autoscaling	Autoscaler berbasis CPU
Minimal Instance	2
Maksimal Instance	10
CPU Utilization Target	60%)
Cool-down Period	60 detik
Scale-in Policy	Kurangi 1 *instance* jika CPU < 30% selama 5 menit

Strategi Distribusi Beban, Strategi yang digunakan dalam load balancing pada penelitian ini adalah *Least Connection Load Balancing*, yang bekerja dengan prinsip: Setiap permintaan dari pengguna akan diarahkan ke server dengan jumlah koneksi paling sedikit untuk memastikan distribusi beban yang merata. Jika terjadi lonjakan lalu lintas pengguna, autoscaler akan menambah *instance server* sehingga beban tetap terdistribusi secara optimal. *Caching* melalui *Cloud* CDN juga diterapkan untuk mengurangi beban *request* ke *server backend* dengan menyajikan konten statis dari *edge servers.*

3.3. Ringkasan Alur Kerja

Pengguna mengakses *platform* melalui web, yang permintaannya diterima oleh HTTPS *Load Balancer. Load Balancer* mendistribusikan permintaan ke beberapa *instance Compute Engine* yang menjalankan aplikasi *Moodle*, memastikan beban didistribusikan merata. *Instance Compute Engine* menggunakan *Cloud SQL* untuk mengakses

data pengguna, kursus, dan konten pembelajaran. Untuk meningkatkan kinerja, *Cloud Memorystore Redis* digunakan sebagai *cache* untuk menyimpan data sesi dan sering diakses. *Cloud Filestore* menyediakan penyimpanan bersama untuk data *Moodle* yang dapat diakses oleh semua *instance*. Tugas konversi dokumen dijalankan oleh *Unoconv Job* di *Compute Engine*, memungkinkan pengguna untuk bekerja dengan berbagai format dokumen dalam *platform. Cloud Armor* memastikan keamanan dengan memfilter dan melindungi lalu lintas dari ancaman eksternal.

Arsitektur ini dirancang dengan mempertimbangkan kebutuhan skalabilitas, kinerja, untuk *platform* pembelajaran *online*. Dengan menggunakan berbagai layanan GCP seperti *Compute Engine*, *Cloud SQL*, *Redis*, *Filestore*, dan *Load Balancer*, sistem ini mampu memberikan pengalaman pengguna yang cepat, andal, dan aman, yang sangat penting untuk mendukung proses pembelajaran digital di UHAMKA.

3.4 Implmentasi Perancangan

Berdasarkan rancangan arsitektur yang telah dibuat di atas kemudian diimplemetasikan pembangunan *system online learning* berbasis *moodle* dalam *platform* GCP, dengan menggunakan *resource* yang tersedia di GCP. Adapun Implementasi s*ystem* tersebut meliputi beberapa *resource* yang telah tersedia dalam *platform* GCP (*Google Cloud Platform*) dengan Spesifikasi dipilih sesuai kebutuhan perancangan.

1. *Compute Engine* Untuk Aplikasi *Moodle. Resource Compute Engine* (*Moodle* APP) yang diimplementasikan dapat dilihat pada tabel 1. berisi spesifikasi sumber daya yang digunakan untuk mendukung implementasi *Moodle* sebagai *platform* pembelajaran *online* di *Google Cloud Platform* (GCP).

Tabel 5. *Resource Compute Engine* (*Moodle* APP)

CPU	RAM	Persistent Disk	Image	Region
4	15 GB	30	Ubuntu 18.04	Asia-southeast 2-a

CPU: 4 vCPU, Jumlah inti prosesor virtual (vCPU) yang dialokasikan untuk menjalankan aplikasi *Moodle*. vCPU adalah sumber daya komputasi yang digunakan untuk memproses operasi dan permintaan pengguna. RAM: 15 GB, Memori yang dialokasikan untuk server aplikasi *Moodle*. RAM yang cukup besar diperlukan untuk menangani beban kerja aplikasi, termasuk permintaan pengguna dan pemrosesan data. *Persistent Disk*: 30 GB, Kapasitas penyimpanan yang digunakan untuk menyimpan data aplikasi dan file sistem. Disk persisten ini digunakan agar data tetap tersedia meskipun instance dimatikan atau di-*restart*. *Image*: Ubuntu 18.04, Sistem operasi yang digunakan pada *instance Compute Engine* ini adalah Ubuntu 18.04, salah satu distribusi Linux yang populer dan stabil, ideal untuk menjalankan aplikasi web seperti *Moodle*. *Region*: asia-*southeast2*-a, Lokasi fisik dari sumber daya *Compute Engine*. Pemilihan *region* di Asia Tenggara memastikan latensi rendah bagi pengguna yang berada di wilayah tersebut, yang penting untuk responsivitas aplikasi.

2. *Cloud SQL* untuk *database Moodle*. Spesifikasi *Cloud SQL (Moodle Database)* yang diimplementasikan dapat dilihat pada tabel 2. berisi spesifikasi sumber daya yang digunakan untuk mendukung implementasi *Moodle* sebagai platform pembelajaran online di *Google Cloud Platform* (GCP).

Tabel 6. *Cloud SQL (Moodle Database)*

CPU	RAM	SSD	Type	Version	Region
2	7.5 GB	70 GB (Auto Increase)	Ubuntu 18.04	5.7	Asia-southeast 2-a

CPU: 2 vCPU, Jumlah inti prosesor virtual yang dialokasikan untuk basis data *Moodle*. *Cloud SQL* menggunakan prosesor ini untuk menangani kueri dan transaksi basis data. RAM: 7.5 GB, Memori yang dialokasikan untuk menjalankan *database MySQL* yang digunakan oleh Moodle. RAM ini membantu dalam meningkatkan kinerja kueri dan manajemen data, terutama untuk *caching*. SSD: 70 GB (*Auto Increase*), Penyimpanan berbasis *Solid State Drive* (SSD) yang digunakan untuk menyimpan data basis data *Moodle*. Penyimpanan ini dilengkapi dengan fitur *auto increase*, yang berarti kapasitas penyimpanan akan meningkat secara otomatis sesuai kebutuhan, tanpa harus mematikan *instance. Type: MySQL*, Jenis basis data yang digunakan adalah *MySQL*, salah satu sistem manajemen basis data relasional yang terkenal dengan keandalan dan kinerjanya. *Version*: 5.7, Versi *MySQL* yang digunakan adalah 5.7, yang dikenal stabil dan mendukung berbagai fitur yang diperlukan oleh aplikasi seperti *Moodle. Region*: asia-southeast2-b, Lokasi fisik dari sumber daya *Cloud SQL*. Menempatkan database di *region* yang sama atau dekat dengan server aplikasi membantu mengurangi latensi akses *database*.

3. *Cloud Memorystore Redis* untuk *Moodle* MUC. Spesifikasi *Cloud Memorystore Redis* (*Moodle* MUC) yang diimplementasikan dapat dilihat pada tabel 3. berisi spesifikasi sumber daya yang digunakan untuk mendukung implementasi *Moodle* sebagai platform pembelajaran *online* di *Google Cloud Platform* (GCP).

Tabel 7. *Cloud Memorystore Redis* (*Moodle* MUC)

Tier	Region	Capacity	Redis Version
Basic	Asia-southeast 2-b	1 GB	4.0

Tier: *Basic, Tier "Basic"* berarti ini adalah konfigurasi sederhana yang cocok untuk beban kerja yang tidak memerlukan ketersediaan tinggi atau *failover* otomatis. *Region*: asia-*southeast2*-b, Lokasi fisik dari *Redis instance* yang sama dengan lokasi *database* untuk meminimalkan *latensi*. *Capacity*: 1 GB, Kapasitas memori yang dialokasikan untuk Redis, yang digunakan oleh *Moodle* untuk *caching* (*Memory User Cache*, MUC). *Caching* membantu meningkatkan kinerja aplikasi dengan menyimpan data yang sering diakses di *memori*. *Redis Version*: 4.0, Versi Redis yang digunakan adalah 4.0, yang mendukung berbagai fitur *caching* dan penyimpanan data di memori yang cepat dan efisien.

4. *Filestore* untuk *Moodledata*. Spesifikasi *Filestore* (*Moodledata*) yang diimplementasikan dapat dilihat pada tabel 4. berisi spesifikasi sumber daya yang digunakan untuk mendukung implementasi Moodle sebagai *platform* pembelajaran *online* di *Google Cloud Platform* (GCP).

Tabel 8. *Filestore (Moodledata)*

Tier	Region	Capacity
Basic	Asia-southeast2-b	1 GB

*Tier: BASIC*_HDD, *Tier* penyimpanan "BASIC_HDD" berarti menggunakan penyimpanan *hard disk drive* (HDD) yang lebih murah dan cukup untuk beban kerja penyimpanan file yang besar tetapi tidak membutuhkan kecepatan tinggi seperti SSD. *Region*: asia-*southeast2*-b, Lokasi fisik dari *Filestore*, yang sama dengan *database* dan *Redis* untuk efisiensi akses. *Capacity*: 1 TB, Kapasitas penyimpanan 1 *Terabyte* yang dialokasikan untuk menyimpan *data Moodle* (*Moodledata*), yang mencakup file pengguna, *cache*, dan data lainnya yang tidak disimpan dalam *database*.

5. Implementasi *Load Balancer*. Spesifikasi *Load Balancer* yang di diimplementasikan dapat dilihat pada tabel 5. berisi spesifikasi sumber daya yang digunakan untuk mendukung implementasi *Moodle* sebagai *platform* pembelajaran *online* di *Google Cloud Platform* (GCP).

Tabel 9. *Load Balancer*

No.	Name	Mode	Protocol	Description
1	bb-moodel-dev-http	HTTPS *Load Balancer*	HTTP	Untuk *Redirect* HTTP ke HTTPS
2	lb-moodel-dev	HTTPS *Load Balancer*	HTTPS	*Handle Traffic*

Load Balancer, No 1 - *Name*: lb-moodle-dev-http. Mode: HTTPS *Load Balancer*, *Load Balancer* yang digunakan untuk menangani lalu lintas HTTP dan mengalihkan (*redirect*) ke HTTPS. *Protocol*: HTTP, *Protokol* ini digunakan untuk menangani permintaan HTTP dari pengguna. *Description*: Untuk *redirect* HTTP ke HTTPS, *Load Balancer* ini dirancang khusus untuk mengalihkan lalu lintas HTTP ke HTTPS, memastikan bahwa semua komunikasi aman. No 2 - Name: lb-moodle-dev. Mode: HTTPS *Load Balancer*, *Load Balancer* ini menangani lalu lintas HTTPS langsung, memastikan semua data dikirimkan dengan aman. *Protocol*: HTTPS, Protokol yang digunakan untuk menangani permintaan HTTPS dari pengguna. *Description*: *Handle traffic Load Balancer* ini digunakan untuk menangani seluruh lalu lintas ke aplikasi *Moodle*, mendistribusikan beban ke beberapa *server* untuk memastikan kinerja optimal.

3.4. Analisa Hasil Implementasi

Analisa dilakukan dengan melakukan pengamatan dari unjuk kerja *system* melalui *dashboard* yang disediakan oleh GCP. Pengamatan dilakukan pada saat pelaksanaan ujian akhir semester pada tanggal 15 Juli 2024. Pada tanggal tersebut dilakukan ujian secara *online* pada jam 08.00 dan 10.00. Karena ujian *online* dilakukan secara serempak oleh beberapa kelas di Fakultas Teknik Industri dan Informatika UHAMKA, maka pada jam 08.00 dan jam 10.00 terjadi lonjakan *user* yang mengakses *online learning* secara bersamaan (*concurrent*). Dengan demikian unjuk kerja *system load balancing* bisa diamati pada jam tersebut.

Pengujian dilakukan untuk mengukur bagaimana HTTPS Load Balancer pada GCP mendistribusikan lalu lintas pengguna ke beberapa instance server. Simulasi dilakukan dengan 1.000 hingga 5.000 pengguna serentak yang mengakses *platform Moodle* dalam berbagai skenario penggunaan. Hasil distribusi beban menunjukkan bahwa: Setiap permintaan diarahkan ke server dengan koneksi aktif paling sedikit (*strategi Least Connection Load Balancing*).

Waktu respons rata-rata tetap stabil di kisaran 200-300 ms, meskipun jumlah pengguna meningkat hingga 5.000. *Health check* otomatis mengalihkan trafik dari instance yang gagal, memastikan hanya server yang sehat melayani pengguna. Setiap permintaan diarahkan ke server dengan koneksi aktif paling sedikit (*strategi Least Connection Load Balancing*).

Waktu respons rata-rata tetap stabil di kisaran 200-300 ms, meskipun jumlah pengguna meningkat hingga 5.000. *Health check* otomatis mengalihkan trafik dari instance yang gagal, memastikan hanya server yang sehat melayani pengguna.

Pengujian *Autoscaler, Autoscaler* diuji untuk melihat efektivitasnya dalam menyesuaikan jumlah *instance* berdasarkan beban kerja. Pada beban rendah (CPU < 30%), *autoscaler* hanya mempertahankan 2 *instance* VM. Saat beban tinggi (CPU > 60%), *autoscaler* menambah *instance* baru dalam waktu kurang dari 30 detik. Saat beban kembali turun, *autoscaler*

mengurangi jumlah *instance*, mengoptimalkan efisiensi biaya.

Tabel 10. Pengujian *Load Balancer*

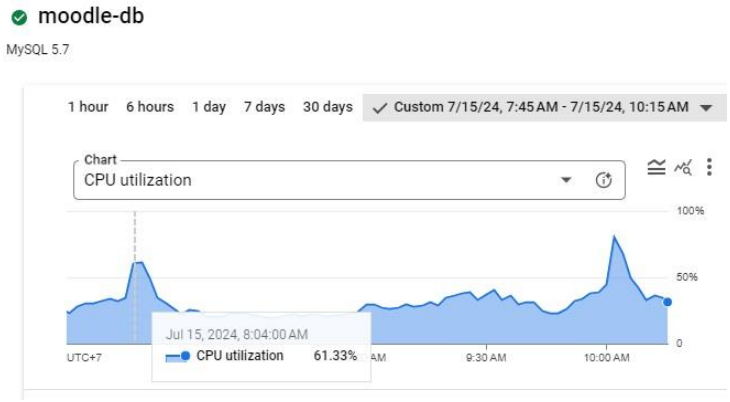
Jumlah Pengguna	VM Aktif	Distribusi Trafik per VM (%)	Waktu Respons Server (ms)
1.000	2	50% - 50%	210
2.500	4	25% - 25% - 25% - 25%	250
5.000	6	16% - 16% - 16% - 16% - 16% - 20%	300

Pengujian Performa dari Sisi Pengguna Selain pengujian pada sisi server, dilakukan juga pengujian waktu respons dari sisi pengguna untuk menilai pengalaman penggunaan *Moodle* setelah *load balancing* dan *autoscaling* diterapkan. Pengujian dilakukan dengan mengukur waktu muat halaman utama, halaman kursus, dan pengiriman UAS dari berbagai lokasi pengguna. Hasil pengujian performa dari sisi klien menunjukkan: Waktu muat halaman utama *Moodle* stabil di bawah 1 detik, bahkan saat beban tinggi. Waktu muat halaman kursus tetap di bawah 1,5 detik, tanpa adanya lonjakan signifikan saat *autoscaler* menambahkan *instance* baru. Proses pengiriman UAS (file upload) tetap responsif, dengan waktu unggah rata-rata di bawah 2 detik. Dari hasil pengujian ini, dapat disimpulkan bahwa:

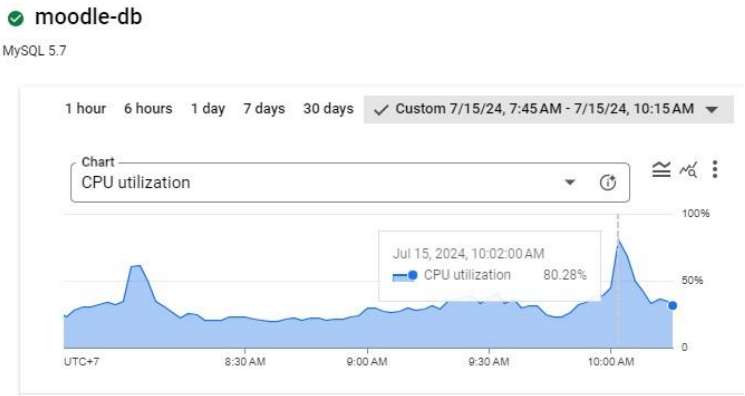
Load balancing memastikan waktu respons tetap stabil dari sisi pengguna, meskipun terjadi peningkatan jumlah pengguna. *Autoscaling* dapat menyesuaikan jumlah VM tanpa menyebabkan gangguan pada pengguna akhir. Pengguna tetap mendapatkan pengalaman pembelajaran yang lancar, bahkan saat sistem menangani 5.000 pengguna serentak.

1. Pengamatan *Moodle Data Base*. Pengamatan unjuk kerja *moodle data base* yang pertama dilakukan pada jam 08.00 tanggal 15 Juli 2024. Pada jam ini dimulai jadwal pertama dari pelaksanaan ujian akhir semester yang diikuti secara serempak oleh semua mahasiswa. Pada jam tersebut terjadi lonjakan user yang mengakses secara bersama *(concurrent)* terhadap *system online learning* yang di bangun menggunakan *loadbalancing* berbasis GCP. Hasil Pengamatan bisa dilihat dalam gambar 3 dan gambar 4.
   Pada gambar 3 dapat diamati bahwa lonjakan penggunaan *data base* pada aplikasi *moodle* yang di rancang terjadi pada jam 08:04, dimana pemakaian *moodle database* mencapai 61, 33 %. Hal ini sesuai dengan jadwal dimulainya ujian akhir semester pada jam pertama.

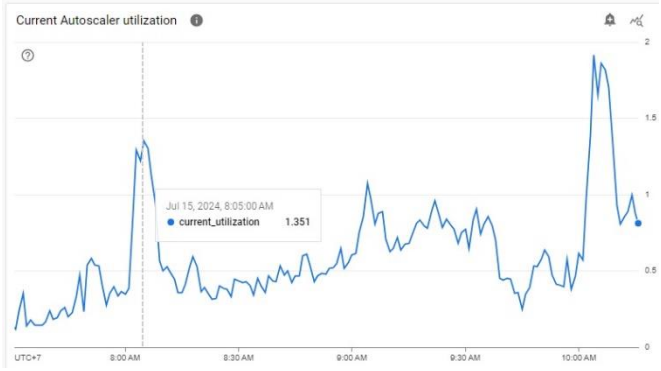
Pada gambar 4 dapat diamati Pada awal periode pengamatan, pemanfaatan CPU tampak stabil dengan beberapa fluktuasi kecil, di mana nilai pemanfaatan CPU berkisar antara 10% hingga 30%. Lonjakan penggunaan *moodle data base* juga terjadi pada pada jam 10:02, dimana penggunaan *moodle data base* mencapai 80,28 %. Hal ini sesuai dengan dimulainya jadwal ujian akhir semester di jam kedua.



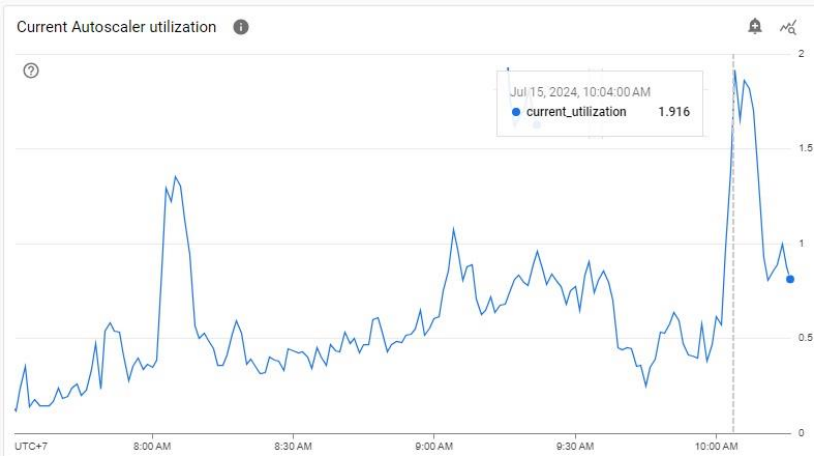
Gambar 3. Penggunaan *Moodle data base* pada jam pertama UAS



Gambar 4. Penggunaan *Moodle Data Base* pada jam kedua UAS.



Gambar 5. Pengamatan Penggunaan *autoscaler* pada jam pertama UAS.



Gambar 6. Pengamatan Penggunaan *Autoscaler* pada jam kedua UAS.

2. Pengamatan Unjuk Kerja *Auto Scaling*. Pengamatan unjuk kerja *auto scaling* dilakukan dengan mengamati parameter penggunaan *autoscaling* (*current aoutoscaler utilization*) yang tersedia pada GCP. Fokus pengamatan dilakukan pada jam 08:00 dan 10:00, tanggal 15 Juli 2024, Dimana pada tanggal tersebut dilaksanakan Ujian Akhir Semester 2024. Hasil pengamatan dapat dilihat pada gambar 4 dan 5. Pada gambar 4. Dapat diamati Sumbu X (*Horizontal*): Menampilkan waktu dalam format 24 jam (UTC+7), mulai dari sekitar pukul 8:00 hingga 10:00 pagi. Sumbu Y (Vertikal): Menunjukkan tingkat utilisasi *autoscaler*, dengan nilai mulai dari 0 hingga 2. Nilai ini mengindikasikan seberapa banyak sumber daya yang digunakan relatif terhadap target yang diatur untuk *autoscaling*. Garis Biru: Menunjukkan nilai aktual dari *current_utilization* pada berbagai waktu. Setiap titik pada garis ini menunjukkan tingkat pemanfaatan sumber daya pada saat tertentu. bahwa lonjakan penggunaan *auto scaling* pertama kali terjadi pada jam 08:05, dimana penggunaan *autoscaling* mencapai level 1,351. Hal ini bersamaan dengan jadwal pelaksanaan ujian akhi semester pada jam pertama.

Pada gambar 5 dapat dilihat lonjakan penggunaan *auto scaler* juga terjadi pada jam 10:04, dimana penggunaan *auto scaler* melonjak sampai level 1,916. Hal ini terjadi seiring dengan dimulainya pelaksanaan ujian akhir semester pada jadwal yang ke dua. Setelah puncak sekitar pukul 10:00, utilisasi mulai menurun dan menunjukkan stabilisasi mendekati nilai 1. Stabilitas ini mengindikasikan bahwa *autoscaler* berhasil menyesuaikan jumlah *instance* dengan kebutuhan beban kerja, menjaga performa optimal aplikasi. Grafik menunjukkan bahwa *autoscaler* berfungsi dengan baik, menambah atau mengurangi kapasitas sesuai dengan fluktuasi beban kerja, memastikan bahwa aplikasi tetap responsif meskipun ada lonjakan permintaan.

Dari hasil pengamatan di atas dapat diketahui bahwa lonjakan *user* yang mengakses sistem *Online Learning* yang dirancang mengakibatkan peningkatan penggunaan *data base*. Lonjakan tersebut bisa diantisipasi dengan *system loadbalancing*, hal ini bisa diamati dari parameter *current autoscaler utilization* yang sudah melebihi level 1 yang berarti bahwa mesin *server* yang bekerja menangani system sudah meningkat menjadi lebih dari 1 mesin *server* (2 mesin *server*). Grafik ini menggambarkan respons dinamis dari infrastruktur GCP terhadap beban yang tidak tetap dari *platform* pembelajaran *online*, memastikan pengalaman pengguna yang optimal dan mengelola sumber daya secara efisien

4. KESIMPULAN

Hasil penelitian ini menunjukkan bahwa implementasi *Load Balancing* pada *Google Cloud Platform* memberikan dampak positif yang signifikan terhadap kinerja, keandalan, dan skalabilitas *platform* pembelajaran *online*. Dengan *Load Balancer*, *platform* mampu menangani lonjakan pengguna secara efisien, mengurangi latensi, meningkatkan *throughput*, dan menjaga ketersediaan sistem. Selain itu, kemampuan untuk secara dinamis menyesuaikan kapasitas *server* melalui *Auto-Scaling* memastikan bahwa *platform* dapat tetap responsif meskipun dalam kondisi beban kerja yang fluktuatif. Namun, penting untuk dicatat bahwa penggunaan *Load Balancing* juga memerlukan optimasi di berbagai tingkatan, termasuk pada tingkat aplikasi dan basis data, untuk menghindari *bottleneck* yang dapat mengurangi efektivitas keseluruhan. Selain itu, meskipun ada peningkatan biaya, dengan strategi optimasi yang tepat, biaya operasional dapat diminimalkan tanpa mengorbankan kinerja. Secara keseluruhan, penelitian ini membuktikan bahwa *Load Balancing* merupakan komponen krusial dalam membangun *platform* pembelajaran *online* yang andal dan skalabel di era digital saat ini. Implementasi yang baik dapat memberikan pengalaman pengguna yang lebih baik dan menjaga kontinuitas proses pembelajaran, terutama dalam situasi di mana jumlah pengguna dapat meningkat secara tidak terduga.

DAFTAR PUSTAKA

AGORMEDAH, E. K., ADU HENAKU, E., AYİTE, D. M. K., & APORİ ANSAH, E. 2020. Online Learning in Higher Education during COVID-19 Pandemic: A case of Ghana. *Journal of Educational Technology and Online Learning*, *3*(3). https://doi.org/10.31681/jetol.726441

ALIM, E. S., & JIN, H. 2017. Deployment of cloud computing for higher education using google apps. 2017 2nd International Conferences on Information Technology, Information Systems and Electrical Engineering (ICITISEE), 73–77. https://doi.org/10.1109/ICITISEE.2017.8285563

ALIZADEH, M., MEHRAN, P., KOGUCHI, I., & TAKEMURA, H. 2019. Evaluating a blended course for Japanese learners of English: why Quality Matters. International Journal of Educational Technology in Higher Education, 16(1). https://doi.org/10.1186/s41239-019-0137-2

APOKO, T. W., & SYA'BAN, M. B. A. 2022. The impact of online learning implementation on satisfaction, motivation, quality of the learning process, and student learning outcomes. AL-ISHLAH: Jurnal Pendidikan, 14(4). https://doi.org/10.35445/alishlah.v14i4.2096

ARDIANSYAH, L., & AWALLUDIN, S. A. 2023. Implementation of the K-Mean Algorithm to Determine the Level of Student Satisfaction with the Online Learning Uhamka System (OLU). JURNAL PEMBELAJARAN DAN MATEMATIKA SIGMA (JPMS), 9(1). https://doi.org/10.36987/jpms.v9i1.4121

BOULMIER, A., ABDENNADHER, N., & CHOPARD, B. 2022. Optimal load balancing and assessment of existing load balancing criteria. Journal of Parallel and Distributed Computing, 169. https://doi.org/10.1016/j.jpdc.2022.07.002

DAFA, N., & TEWU, D. 2023. Evaluasi Pelaksanaan Pembelajaran Jarak Jauh pada Masa Pandemi Covid-19 di SMP Negeri 2 Kupang Barat. Journal on Education, 6(1). https://doi.org/10.31004/joe.v6i1.3026

ENDY SJAIFUL ALIM. 2017. Information Technology (IT) Architecture Based on Cloud Computing for Muhammadiyah Higher Education Institutions (HEIs). US-China Education Review A, 7(4). https://doi.org/10.17265/2161-623x/2017.04.004

GUMASING, M. J. J., & CASTRO, F. M. F. 2023. Determining Ergonomic Appraisal Factors Affecting the Learning Motivation and Academic Performance of Students during Online Classes. Sustainability (Switzerland), 15(3). https://doi.org/10.3390/su15031970

KAUR, M. 2020. Changes in Content of Online Teaching in pre and during COVID-19 Era: A Comparative Analysis in Private School Setup in India. JOURNAL OF TEACHER EDUCATION AND RESEARCH, 15(02). https://doi.org/10.36268/jter/15203

KHAN, A., & ALI, M. 2023. Three-Phase Load Balancing in Distribution Systems Using Load Sharing Technique †. Engineering Proceedings, 46(1). https://doi.org/10.3390/engproc2023046018

MISHRA, S. K., SAHOO, B., & PARIDA, P. P. 2020. Load balancing in cloud computing: A big picture. In Journal of King Saud University - Computer and Information Sciences (Vol. 32, Issue 2). https://doi.org/10.1016/j.jksuci.2018.01.003

MULYANI, A., KURNIADI, D., & PUTRI, M. H. 2023. Analisis Penerimaan Learning Management System Institut Teknologi Garut Menggunakan Technology Acceptance Model. Jurnal Teknologi Informasi Dan Ilmu Komputer (JTIIK), 10(4), 843–850. https://doi.org/10.25126/jtiik.2023106618

NANCY, J. J., TAMIL MANI, S., ROHITH, S., SARANRAJ, S., & VIGNESWARAN, T. (2020). Load Balancing using Load Sharing

Technique in Distribution System. 2020 6th International Conference on Advanced Computing and Communication Systems, ICACCS 2020. https://doi.org/10.1109/ICACCS48705.2020.9074304

NUJID, M. M., & THOLIBON, D. A. 2023. Evaluation of Learners' Academic Performance in Teaching and Learning Civil Engineering During the COVID-19 Pandemic. International Journal of Engineering Pedagogy, 13(3). https://doi.org/10.3991/ijep.v13i3.30147

QAZI, A., QAZI, J., NASEER, K., ZEESHAN, M., QAZI, S., ABAYOMI-ALLI, O., SAID AHMAD, I., DARWICH, M., ALI TALPUR, B., HARDAKER, G., NASEEM, U., YANG, S., & HARUNA, K. 2021. Adaption of distance learning to continue the academic year amid COVID-19 lockdown. Children and Youth Services Review, 126. https://doi.org/10.1016/j.childyouth.2021.106038

RIZKIAWAN, M. A., RAMZA, H., & ALIM, E. S. 2023. Sistem Informasi Pencatatan Aset Dan Peminjaman Barang Menggunakan Metode Pengembangan Agile Pada BPTI UHAMKA. Journal of Scientech Research and Development, 5(2). https://idm.or.id/JSCR/in

RIZKIAWAN, M. A., SIREGAR, M. T., RAMZA, H., & ALIM, E. S. 2023. Designing an Information System for Recording Assets (Equipment) and Lending of Goods UHAMKA Information Technology Development Agency. International Journal of Advances in Engineering and Management (IJAEM), 5(6), 973–979. https://doi.org/10.35629/5252-0506973979

ROUT, S. K., RAVINDRA, J. V. R., MEDA, A., MOHANTY, S. N., & KAVIDIDEVI, V. 2023. A Dynamic Scalable Auto-Scaling Model as a Load Balancer in the Cloud Computing Environment. EAI Endorsed Transactions on Scalable Information Systems, 10(5). https://doi.org/10.4108/eetsis.3356

SOFI-KARIM, M., BALI, A. O., & RACHED, K. 2023. Online education via media platforms and applications as an innovative teaching method. Education and Information Technologies, 28(1). https://doi.org/10.1007/s10639-022-11188-0

TALEB, N., & MOHAMED, E. A. 2020. Cloud computing trends: A literature review. In Academic Journal of Interdisciplinary Studies (Vol. 9, Issue 1). https://doi.org/10.36941/ajis-2020-0008

ZHOU, J., LILHORE, U. K., POONGODI, M., HAI, T., SIMAIYA, S., JAWAWI, D. N. A., ALSEKAIT, D., AHUJA, S., BIAMBA, C., & HAMDI, M. 2023. Comparative analysis of metaheuristic load balancing algorithms for efficient load balancing in cloud computing. Journal of Cloud Computing, 12(1). https://doi.org/10.1186/s13677-023-00453-3