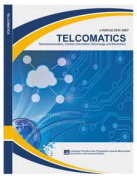
Available online at journal.uib.ac.id/index.php/telcomatics

TELCOMATICS

ISSN (online) 2541-5867



Ketersediaan Tinggi Website menggunakan Penskalaan Otomatis & Penyeimbang Beban AWS

Haeruddin[1*], Andik Yulianto[2], Stefanus Eko Prasetyo[1], Gautama Wijaya Wijaya[1], Dominggo Givarel[1]

[1]*Prodi Teknologi Informasi, Fakultas Ilmu Komputer, Universitas Internasional Batam, Jalan Gajah Mada, Baloi Sei Ladi, Batam, Indonesia*
[2]*Prodi Sistem Informasi, Fakultas Ilmu Komputer, Universitas Internasional Batam, Jalan Gajah Mada, Baloi Sei Ladi, Batam, Indonesia*

ARTICLE INFO

Keywords:
Website, Autoscaling, Load Balancing, VPS, EC2, AWS

Received: June 10, 2025
Revised: July 20, 2025
Accepted: July 30, 2025

***Corresponding author:**
E-mail: haerudin@uib.ac.id (Haerudin)

DOI: 10.37253/telcomatics.v10i1.10951

ABSTRACT

The rapid development of information technology requires digital service systems to have high performance and optimal availability. One of the main challenges in managing web-based services is the system's ability to deal with an unpredictable surge in the number of users. If not anticipated, this can cause a decrease in performance and even detrimental downtime. This study aims to analyze and implement the concept of Auto Scaling and load balancing in an effort to improve the performance and availability of web-based services. Auto Scaling functions to automatically adjust the number of server resources according to workload needs, while load balancing plays a role in distributing network traffic evenly to several servers. The research method used is an experiment, by implementing and testing service performance before and after the implementation of Auto Scaling and load balancing. The test results show that the combination of the two technologies is able to increase the speed of service response, reduce server load, and maintain optimal service availability. This research is expected to be a reference in the development of reliable and efficient cloud-based systems.

I. PENDAUHULUAN

Perkembangan teknologi informasi dan komunikasi yang sangat pesat telah mendorong berbagai sektor industri untuk beradaptasi dengan sistem berbasis digital. Salah satu aspek penting dalam pengelolaan sistem digital adalah ketersediaan layanan (*availability*) dan performa sistem yang stabil, khususnya bagi layanan berbasis web dan aplikasi yang memiliki trafik pengguna yang dinamis. Dalam implementasi layanan digital, tantangan utama yang sering dihadapi adalah ketidakpastian jumlah permintaan layanan yang dapat berubah sewaktu-waktu. Lonjakan trafik pengguna yang terjadi secara tiba-tiba dapat menyebabkan *server* mengalami kelebihan beban, sehingga berdampak pada menurunnya performa layanan bahkan hingga terjadinya *downtime*. Kondisi ini tentu sangat merugikan, terutama bagi layanan yang bersifat kritis dan bergantung pada ketersediaan sistem sepanjang waktu.

Untuk mengatasi permasalahan tersebut, dibutuhkan mekanisme yang mampu mengelola sumber daya komputasi secara dinamis dan efisien. Salah satu solusi yang banyak digunakan dalam infrastruktur modern, khususnya pada lingkungan *cloud computing*, adalah *Auto Scaling*. *Auto Scaling* merupakan mekanisme yang memungkinkan sistem untuk secara otomatis menambah atau mengurangi jumlah resource (seperti *server virtual*) sesuai dengan kebutuhan beban kerja pada saat tertentu. Dengan demikian, sistem dapat tetap berjalan optimal tanpa mengalami *overload* maupun pemborosan sumber daya. Selain itu, penerapan load balancing menjadi komponen penting dalam arsitektur sistem modern. Load balancing berfungsi untuk mendistribusikan lalu lintas jaringan atau permintaan layanan secara merata ke beberapa *server backend*. Dengan adanya *load balancing*, sistem dapat menghindari terjadinya penumpukan beban pada satu *server*, serta meningkatkan keandalan dan ketersediaan layanan.

Integrasi antara *Auto Scaling* dan load balancing memberikan solusi yang efektif dalam pengelolaan sistem berskala besar, di mana sistem dapat secara otomatis menyesuaikan kapasitas sumber daya sesuai kebutuhan dan mendistribusikan beban secara optimal. Oleh karena itu, penelitian ini dilakukan untuk menganalisis dan mengimplementasikan konsep *Auto Scaling* dan load balancing dalam rangka meningkatkan performa, efisiensi, dan ketersediaan layanan berbasis web. Dalam penelitian ini dapat bermanfaat dalam pengembangan dan menciptakan layanan yang efesien, *scalable*, tahan terhdap lonjakan beban. Serta memberikan panduan dalam penerapan *Auto Scaling* dan load balancing pada lingkungan *Cloud* untuk mengoptimalkan performa aplikasi, meningkatkan kualitas layanan, kestabilan, dan ketersediaan uptime.

Tujuan dari penelitian ini adalah membuat dan mengembangkan website dengan metode *Auto Scaling* yang respontan dan *load ballancing* pada sistem virtual AWS, Menganalisis pengaruh penerapan *Auto Scaling* terhadap penggunaan sumber daya sistem dan performa layanan, Mengkaji efektivitas load balancing dalam mendistribusikan beban trafik pengguna dan mengevaluasi peningkatan performa dan ketersediaan layanan melalui penerapan kombinasi *Auto Scaling* dan load balancing pada sistem *cloud* [1].

II. KAJIAN PUSTAKA

A. Teknologi *Cloud computing*

Cloud computing merupakan model layanan komputasi berbasis internet yang menyediakan sumber daya komputasi

seperti *server*, penyimpanan data, jaringan, dan aplikasi secara fleksibel sesuai kebutuhan. Menurut [2], *cloud computing* memiliki lima karakteristik utama yaitu *on-demand self-service, broad network access, resource pooling, rapid elasticity* dan *measured service*. Model layanan *cloud* terbagi menjadi tiga kategori, yaitu: Infrastructure as a Service (IaaS), Platform as a Service (PaaS), Software as a Service (SaaS). *Cloud computing* menawarkan kelebihan seperti skalabilitas, efisiensi biaya, kemudahan akses, dan fleksibilitas dalam pengelolaan layanan[3].

B. *Load Balancing*

Load balancing adalah teknik distribusi lalu lintas jaringan secara merata ke beberapa *server backend* agar tidak terjadi kelebihan beban pada satu *server* tertentu. Menurut[4], load balancing dapat meningkatkan keandalan, efisiensi, dan kecepatan respon layanan. Metode *load balancing*:

1. Round-Robin. Algoritma round-robin mendistribusikan beban ke semua *server* anggota cluster sehingga masing-masing *server* mendapat beban yang sama dalam waktu yang sama. Round-robin cocok saat *server* anggota cluster memiliki kemampuan proccessing yang sama, jika tidak, beberapa *server* bisa jadi menerima request lebih dari kemampuan processing *server* itu sendiri sedang yang lainnya hanya mendapat beban lebih sedikit dari resource yang dimiliki.
2. Least-connection. Algoritma least-connection melakukan pengiriman request pada *server* anggota cluster, berdasarkan pada *server* mana yang memiliki fewest connections (koneksi paling sedikit).
3. IP Hash: *server* dipilih berdasarkan hasil hash dari alamat IP client.

C. Amazon Web Service

Amazon Web Services (AWS) merupakan salah satu penyedia layanan *cloud computing* terbesar dan paling populer di dunia. AWS menyediakan berbagai layanan komputasi berbasis *cloud* yang dapat digunakan untuk berbagai kebutuhan, mulai dari penyimpanan data, komputasi, basis data, jaringan, keamanan, hingga layanan *machine learning* dan kecerdasan buatan. AWS pertama kali diperkenalkan oleh Amazon pada tahun 2006 dan sejak saat itu telah menjadi pemimpin pasar *cloud computing* global[5].

Amazon Elastic Compute *Cloud* (EC2) merupakan salah satu layanan inti dalam ekosistem Amazon Web Services (AWS) yang menyediakan infrastruktur *server* virtual berbasis *cloud* untuk menjalankan berbagai aplikasi secara fleksibel. EC2 memungkinkan pengguna untuk melakukan provisioning, konfigurasi, dan pengelolaan *server* virtual (disebut instance) secara on-demand, tanpa perlu membeli dan mengelola perangkat keras fisik secara langsung[6].

Amazon Virtual Private *Cloud* (Amazon VPC) adalah layanan yang disediakan oleh Amazon Web Services (AWS) untuk menyediakan lingkungan jaringan virtual yang terisolasi secara logis di dalam *cloud* AWS. Dengan Amazon VPC, pengguna dapat mengontrol lingkungan jaringan *cloud* mereka, termasuk pemilihan rentang alamat IP, pembuatan subnet, konfigurasi tabel routing, hingga pengaturan gateway jaringan[7]. *VPC* memiliki beberapa kelebihan seperti mudah dikelola dengan kendali penuh atas jaringan,mudah diskalakan membrikan kontrol keamanan tingkat tinggi, tersedia dan tahan lama yang berjalan mendukung arsitektur hybrid *cloud*, dan aman berintegerasi dengan layanan aws yang lain.

D. *Cloud computing*

Cloud computing atau komputasi awan merupakan paradigma baru dalam dunia teknologi informasi yang menyediakan sumber daya komputasi (seperti *server*, penyimpanan, database, jaringan, perangkat lunak, dan layanan lainnya) melalui jaringan internet secara fleksibel, elastis, dan sesuai permintaan (on-demand). Komputasi awan memiliki lima karakteristik utama, yaitu (1) on-demand self-service, dimana pengguna dapat mengakses sumber daya tanpa keterlibatan langsung penyedia layanan; (2) broad network access, yang memungkinkan layanan dapat diakses melalui berbagai perangkat dan platform; (3) resource pooling, dimana sumber daya dikonsolidasikan dan dibagikan kepada banyak pengguna secara dinamis; (4) rapid elasticity, yang memungkinkan peningkatan atau penurunan kapasitas secara cepat; dan (5) measured service, yang berarti penggunaan layanan diukur dan dikontrol secara transparan. Manfaat dari *cloud computing* sangat signifikan, antara lain efisiensi biaya karena tidak perlu membeli perangkat keras fisik secara langsung, skalabilitas yang memungkinkan penyesuaian kapasitas sumber daya secara cepat, serta fleksibilitas akses dari berbagai lokasi[9]. Selain itu, penyedia layanan *cloud* umumnya menawarkan keandalan sistem dan dukungan keamanan yang tinggi.

E. *Auto Scalling*

Auto Scaling merupakan mekanisme otomatis dalam lingkungan *cloud computing* yang berfungsi untuk menyesuaikan jumlah sumber daya komputasi berdasarkan beban kerja secara real-time. Tujuan utama dari *Auto Scaling* adalah untuk menjaga kinerja sistem tetap optimal dan efisien dengan cara menambah atau mengurangi instans komputasi (seperti virtual machine atau container) sesuai kebutuhan. Dengan *Auto Scaling*, sistem dapat secara otomatis melakukan penyesuaian tanpa intervensi manual dari administrator, sehingga mendukung efisiensi biaya, fleksibilitas, serta ketersediaan tinggi. Manfaat utama dari *Auto Scaling* mencakup efisiensi biaya—karena sumber daya hanya digunakan ketika diperlukan—serta peningkatan ketersediaan layanan (*high availability*) dan performa sistem secara keseluruhan[10]. Namun, penerapan *Auto Scaling* juga menghadapi tantangan seperti delay dalam proses provisioning instans baru, risiko over-scaling atau under-scaling akibat pengaturan kebijakan yang tidak tepat, serta ketergantungan pada arsitektur aplikasi yang mendukung elastisitas. *Auto Scaling* menjadi komponen penting dalam arsitektur sistem *cloud* modern, terutama ketika dikombinasikan dengan load balancing. Keduanya bekerja bersama-sama untuk memastikan distribusi beban kerja yang optimal dan menjaga sistem tetap responsif meskipun terjadi lonjakan trafik atau perubahan permintaan secara tiba-tiba.

F. *Load Ballancer*

Load Balancer merupakan komponen penting dalam sistem komputasi modern, khususnya dalam arsitektur berbasis *cloud computing*, yang berfungsi untuk

 http://dx.doi.org/10.37253/telcomatics.v10i1.10951

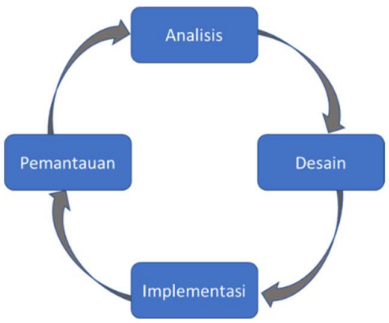
mendistribusikan beban kerja atau lalu lintas jaringan secara merata ke sejumlah *server* atau instans layanan. Tujuan utama dari load balancing adalah untuk menghindari konsentrasi beban pada satu titik, meningkatkan ketersediaan (*availability*), menjaga kestabilan sistem, serta memastikan performa layanan tetap optimal meskipun terjadi lonjakan permintaan. Dalam melakukan distribusi beban, *Load Balancer* menggunakan berbagai algoritma, seperti round robin (mendistribusikan permintaan secara bergilir ke setiap *server*), least connections (memilih *server* dengan koneksi aktif paling sedikit), dan IP hashing (memetakan permintaan berdasarkan alamat IP klien untuk mempertahankan sesi)[11]. Beberapa sistem juga menggunakan weighted round robin atau weighted least connections untuk memberikan proporsi lebih besar kepada *server* dengan kapasitas lebih tinggi. Dalam arsitektur *cloud computing* modern, *Load Balancer* umumnya dikombinasikan dengan fitur *Auto Scaling*, sehingga ketika jumlah instans backend bertambah atau berkurang secara otomatis, *Load Balancer* dapat secara dinamis menyesuaikan distribusi lalu lintas. Kolaborasi ini menghasilkan sistem yang tangguh, adaptif, dan mampu merespons perubahan beban secara efisien, menjadikan *Load Balancer* sebagai elemen fundamental dalam membangun layanan digital berskala besar dan andal.

III. METODE PENELITIAN

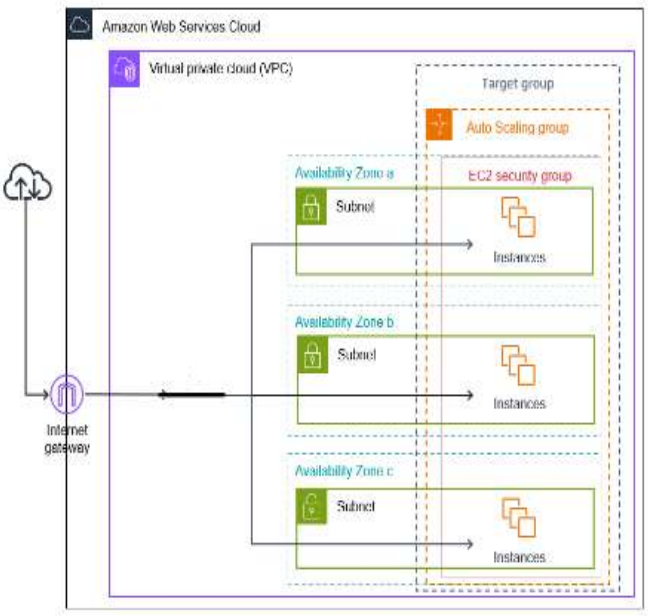
Penelitian ini dilaksanakan menggunakan metode Network, Development, Life, Cycle yang terdiri dari Analisis, Desain, Simulasi Prototyping, Implementasi, Pemantauan, dan Manajemen. Namun pada model siklus tidak semua tahapan digunakan. Tahapan yang di gunakan adalah Analisis, Desain, Implementasi, dan Monitoring. Model siklus hidup pengembangan jaringan seperti yang ditunjukkan pada Gambar 1. Metode ini dipilih karena sesuai untuk mengamati perubahan performa layanan sebelum dan sesudah penerapan *Auto Scaling* dan *load balancing*[8].

A. Analisis

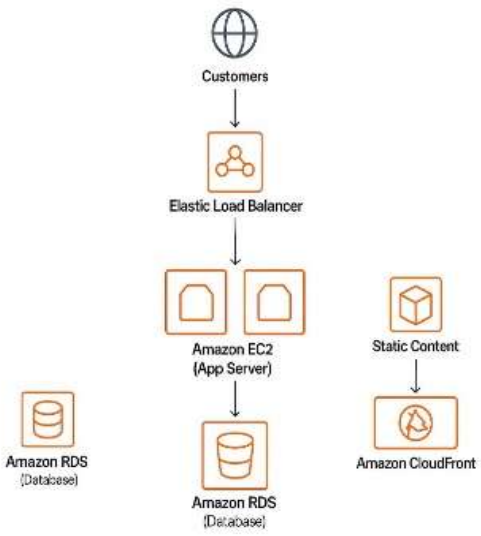
Objek dalam penelitian ini adalah sistem layanan web berbasis *cloud* yang dikembangkan pada platform Amazon Web Services (AWS). Sistem ini dirancang dengan menggunakan beberapa instance *server* yang dapat diskalakan secara otomatis serta dilengkapi dengan *Load Balancer* untuk mendistribusikan beban lalu lintas pengguna. Web *server* yang digunakan berbasis Nginx, sedangkan bahasa pemrograman backend yang digunakan adalah PHP. Topologi auto scaling dalam penelitian ini ditunjukkan pada Gambar 2.



Gambar 1. Metode NDLC



Gambar 2. Topologi *Auto Scalling*



Gambar 3. Topologi Load Ballancer

Topologi *load balancer* ditunjukkan pada Gambar 3. Arsitektur berbasis Amazon Web Services (AWS) ini dirancang untuk mendukung kinerja tinggi, skalabilitas dinamis, serta ketersediaan layanan yang andal. Pengguna (*clients*) mengakses aplikasi melalui perangkat mobile maupun web, di mana setiap permintaan dikirim melalui jaringan internet menuju infrastruktur AWS. Dalam proses ini, layanan Amazon Route 53 (opsional) berfungsi sebagai *Domain Name System (DNS)* yang mengarahkan nama domain ke alamat Elastic Load Balancer (ELB). Komponen ELB berfungsi untuk mendistribusikan beban lalu lintas secara merata ke beberapa *application server* yang dijalankan melalui Amazon EC2 Instances. Dengan demikian, sistem dapat menghindari terjadinya *bottleneck* pada satu server dan meningkatkan tingkat ketersediaan (*high availability*) serta keandalan layanan (*fault tolerance*).

Selanjutnya, server aplikasi berada dalam Auto Scaling Group yang secara otomatis menyesuaikan jumlah instans EC2 berdasarkan beban trafik yang terpantau. Teknik ini disebut horizontal *scaling*. Mekanisme *auto scaling* ini diatur melalui Amazon CloudWatch, yang berfungsi untuk memantau berbagai metrik sistem seperti *CPU utilization*, *latency*, dan *request count*. Apabila terjadi peningkatan permintaan pengguna, CloudWatch memicu penambahan instans EC2 baru guna menjaga kinerja sistem tetap optimal; sebaliknya, ketika beban menurun, jumlah instans akan dikurangi untuk menghemat sumber daya komputasi.

Data operasional seperti informasi pelanggan, menu, dan transaksi pemesanan disimpan dalam Amazon Relational Database Service (RDS), yang merupakan layanan basis data relasional terkelola dengan dukungan mesin seperti MySQL atau PostgreSQL. RDS ditempatkan pada zona aman (*private subnet*) untuk mencegah akses langsung dari publik, sekaligus mendukung replikasi data, *automated backup*, serta *failover* otomatis guna memastikan integritas dan kontinuitas data.

Sementara itu, komponen Amazon CloudFront yang terintegrasi dengan Amazon S3 digunakan untuk menyimpan dan mendistribusikan konten statis seperti gambar menu, file *Cascading Style Sheets (CSS)*, serta skrip *JavaScript*. CloudFront bertindak sebagai *Content Delivery Network (CDN)* yang menyajikan konten dari lokasi *edge* terdekat dengan pengguna, sehingga mengurangi latensi dan mempercepat waktu respons aplikasi di berbagai wilayah geografis. Secara keseluruhan, integrasi antara Route 53, ELB, EC2 Auto Scaling, RDS, CloudFront, S3, dan CloudWatch membentuk arsitektur cloud yang efisien, adaptif, serta mampu menjamin pengalaman pengguna yang optimal dalam sistem pemesanan kafe berbasis web dan mobile.

B. Implementasi

Pada tahap ini, dilakukan implementasi langsung berdasarkan desain arsitektur sistem yang telah dirancang pada tahapan sebelumnya. Seluruh instalasi aplikasi dan konfigurasi fitur pendukung dilakukan dalam lingkungan Amazon Web Services (AWS), guna membangun infrastruktur sistem yang andal, fleksibel, dan skalabel.

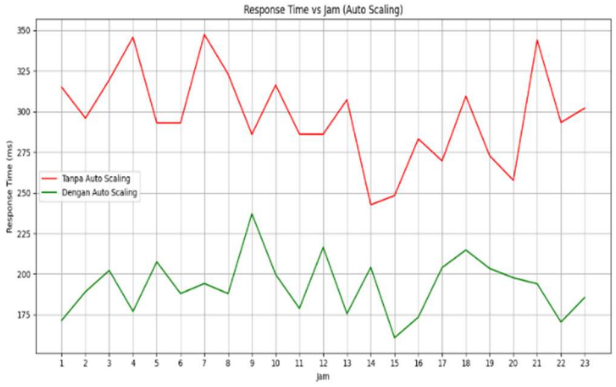
1. Amazon EC2 (Elastic Compute *Cloud*)**:** Digunakan sebagai *server* utama untuk menjalankan aplikasi inti sistem pemesanan, termasuk backend dan web *server*.
2. *Auto Scaling*: Dikonfigurasi untuk menyesuaikan jumlah instans EC2 secara otomatis berdasarkan beban kerja (workload), sehingga sistem dapat menyesuaikan kapasitas secara dinamis sesuai kebutuhan.
3. Elastic *Load Balancer* (ELB): Berfungsi sebagai penyeimbang beban lalu lintas dari pengguna ke beberapa instans EC2, guna memastikan distribusi trafik yang merata dan mencegah terjadinya overload pada salah satu *server*.

C. Monitoring

Tahap monitoring merupakan proses pemantauan terhadap sistem yang telah diimplementasikan sebelumnya. Tujuan utama dari tahap ini adalah untuk memastikan bahwa seluruh komponen sistem berjalan dengan baik dan sesuai dengan perencanaan, serta untuk mendeteksi secara dini apabila terjadi gangguan atau penurunan performa. Pada tahap ini, dilakukan pengujian terhadap kestabilan dan kinerja sistem melalui pemantauan berbagai metrik operasional, seperti penggunaan CPU, memori, jumlah permintaan (requests), dan waktu respons[12]. Pemantauan ini dilakukan secara real-time menggunakan fitur monitoring yang disediakan oleh Amazon *Cloud*Watch, yang secara otomatis menghasilkan grafik visual dari data performa sistem.

IV. Hasil dan Pembahasan

Hasil dan Implementasi yang digunakan diambil dari case ketersediaan tinggi pada sistem Cafe AWS dilakukan dengan menggabungkan fitur *Auto Scaling* dan *Load Balancer* dari Amazon Web Services. Tujuan utama dari implementasi ini adalah untuk memastikan bahwa aplikasi web Cafe AWS tetap tersedia, stabil, dan resilien terhadap lonjakan trafik maupun kegagalan system Cafe AWS merupakan aplikasi berbasis web yang melayani pemesanan makanan dan minuman secara online. Karena adanya potensi lonjakan kunjungan pengguna—misalnya pada jam makan siang atau saat promo—diperlukan solusi infrastruktur yang otomatis dapat menyesuaikan kapasitas layanan berdasarkan beban trafik yang dinamis[13].Berikut hasil implementasi pada system :



Gambar 4. Grafik *Auto Scalling*

A. *Auto Scalling*

Auto Scaling digunakan untuk secara otomatis menambah atau mengurangi jumlah instance EC2 berdasarkan beban kerja Hal ini memungkinkan sistem untuk menghemat biaya saat trafik rendah dan tetap responsif saat trafik meningkat.

Hasil pengujian membuktikan bahwa penerapan *Auto Scaling* memberikan keunggulan signifikan dalam hal kinerja, efisiensi, dan ketahanan sistem terhadap lonjakan trafik. Pada sistem tanpa *Auto Scaling*, jumlah instance EC2 bersifat tetap dan statis. Ketika terjadi peningkatan jumlah pengguna, kapasitas instance menjadi tidak mencukupi untuk melayani permintaan secara optimal. Hal ini menyebabkan peningkatan waktu respon secara drastis, penurunan tingkat keberhasilan permintaan, bahkan kegagalan sistem (crash) saat beban sangat tinggi. Selain itu, sistem tidak dapat menyesuaikan sumber daya dengan kondisi real-time, sehingga memerlukan intervensi manual yang memakan waktu dan berisiko terhadap kelangsungan layanan. Sebaliknya, sistem dengan *Auto Scaling* mampu melakukan penyesuaian kapasitas secara otomatis berdasarkan metrik pemantauan seperti CPU Utilization. Ketika beban meningkat, sistem secara dinamis menambahkan instance baru (scale out), dan ketika beban menurun, sistem akan mengurangi jumlah instance (scale in).

Integrasi ini membuat sistem jauh lebih responsif dan adaptif terhadap perubahan beban kerja, serta memungkinkan penghematan biaya operasional karena sumber daya hanya digunakan sesuai kebutuhan[14]. Dari hasil pengujian performa, sistem dengan *Auto Scaling* menunjukkan waktu respon yang lebih stabil, tingkat keberhasilan permintaan di atas 95% meskipun trafik tinggi, dan kemampuan pulih otomatis dari kondisi overload. *Auto Scaling* juga memudahkan integrasi dengan *Load Balancer*, sehingga distribusi beban dapat dilakukan secara optimal ke semua instance yang aktif. Secara keseluruhan, penerapan *Auto Scaling* pada sistem Cafe AWS terbukti meningkatkan skalabilitas, efisiensi, dan keandalan sistem secara signifikan dibandingkan dengan sistem statis tanpa *Auto Scaling*. Oleh karena itu, fitur ini sangat direkomendasikan untuk aplikasi berbasis *cloud* yang menghadapi fluktuasi trafik dan membutuhkan ketersediaan layanan tinggi secara otomatis dan berkelanjutan.

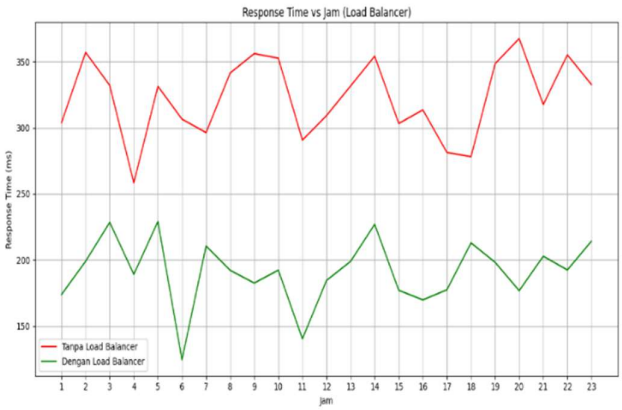
B. *Load Ballancer*

Di lingkungan *cloud* seperti Amazon Web Services (AWS), *Load Balancer* sangat penting untuk memastikan bahwa tidak ada satu *server* pun yang terbebani secara berlebihan, sehingga sistem dapat berjalan lebih stabil, efisien, dan tahan terhadap kegagalan.

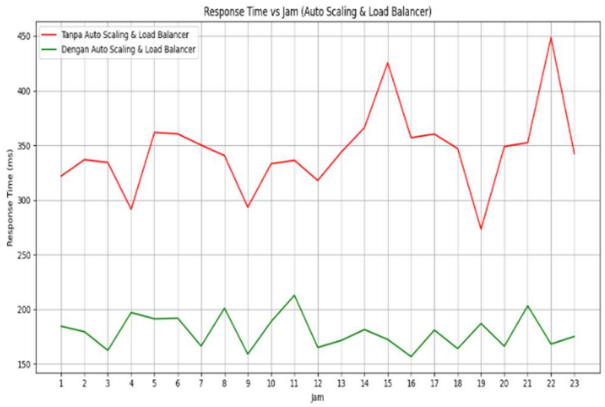
Hasil pengamatan pada Gambar 5 menunjukkan bahwa penggunaan *Load Balancer* memberikan peningkatan yang signifikan terhadap performa, kestabilan, dan ketersediaan layanan. Sistem tanpa *Load Balancer* menunjukkan berbagai keterbatasan. Ketika trafik meningkat, beban hanya tertumpu pada satu atau dua instance secara statis, sehingga menyebabkan overload, peningkatan waktu respon, bahkan downtime saat salah satu instance gagal. Sistem juga tidak memiliki mekanisme pemantauan kesehatan, sehingga trafik tetap diarahkan ke *server* yang tidak aktif atau lambat, memperburuk pengalaman pengguna. Selain itu, pengelolaan dan pemeliharaan *server* dilakukan secara manual, yang tidak efisien dan berisiko pada saat beban berubah-ubah. Sebaliknya, pada sistem yang menggunakan Application *Load Balancer* (ALB) dari AWS, trafik pengguna didistribusikan secara otomatis ke seluruh instance EC2 yang aktif dan sehat. ALB melakukan pemantauan berkelanjutan (health check) dan akan menghentikan pengiriman trafik ke instance yang bermasalah, sehingga menjaga kestabilan sistem[15]. Selain itu, *Load Balancer* bekerja terintegrasi dengan *Auto Scaling*, sehingga mampu menyesuaikan jumlah instance secara otomatis sesuai beban trafik, tanpa perlu intervensi manual. Hasil pengujian menunjukkan bahwa sistem dengan *Load Balancer* mampu menjaga waktu respon tetap rendah, mempertahankan tingkat keberhasilan permintaan di atas 95%, dan menghindari kegagalan sistem bahkan saat trafik tinggi. Integrasi ini juga memungkinkan sistem Cafe AWS untuk tetap efisien dalam penggunaan sumber daya dan hemat biaya saat trafik menurun, berkat skema penskalaan otomatis. Secara keseluruhan, dapat disimpulkan bahwa penggunaan *Load Balancer* merupakan komponen krusial dalam arsitektur sistem modern seperti Cafe AWS. Dibandingkan dengan sistem tanpa *Load Balancer*, pendekatan ini jauh lebih unggul dalam hal keandalan layanan, efisiensi operasional, dan kepuasan pengguna, khususnya pada kondisi trafik yang tidak stabil atau mendadak meningkat.

C. *Auto Scalling* dan *Load Ballancer*

Dalam pengujian dan implementasi sistem Cafe AWS, dilakukan analisis perbandingan terhadap empat kondisi sistem: tanpa *Auto Scaling* dan *Load Balancer*, hanya dengan *Auto Scaling*, hanya dengan *Load Balancer*, serta kombinasi keduanya. Hasil dari studi ini ditampilakn pada Gambar 6 menunjukkan bahwa penggunaan *Auto Scaling* dan *Load Balancer* secara terintegrasi memberikan kinerja terbaik, efisiensi tertinggi, dan tingkat ketersediaan sistem yang optimal.



Gambar 5. Grafik *Load Ballancer*



Gambar 6. Grafik *Auto Scalling* dan Load Ballancer

Pada sistem Cafe AWS yang dioperasikan tanpa dukungan *Auto Scaling* dan *Load Balancer*, ditemukan berbagai keterbatasan yang berdampak langsung pada kinerja dan keandalan sistem. Sistem ini bersifat statis, di mana jumlah instance EC2 tetap dan semua trafik masuk diarahkan secara langsung ke satu atau beberapa instance tanpa adanya distribusi beban otomatis. Ketika terjadi lonjakan trafik, seperti pada waktu sibuk atau saat promosi online, instance yang ada dengan cepat mengalami kelebihan beban (overload), mengakibatkan peningkatan waktu respon, penurunan tingkat keberhasilan permintaan, hingga kegagalan layanan (downtime)[16]. Selain itu, sistem ini tidak memiliki mekanisme pemantauan kesehatan (*health check*), sehingga trafik tetap diarahkan ke *instance* yang bermasalah. Tidak adanya penskalaan otomatis juga mengharuskan intervensi manual ketika beban meningkat, yang tentu tidak praktis dan memperbesar risiko kehilangan pelanggan. Dari hasil pengamatan, sistem tanpa *Auto Scaling* dan *Load Balancer* tidak cocok digunakan untuk aplikasi web yang memiliki pola

trafik dinamis dan membutuhkan ketersediaan tinggi. Sebaliknya, saat sistem Cafe AWS diimplementasikan dengan *Auto Scaling* dan *Load Balancer*, performanya menunjukkan peningkatan yang sangat signifikan. *Auto Scaling* memungkinkan sistem secara otomatis menambah atau mengurangi jumlah instance EC2 berdasarkan beban kerja. Ketika trafik meningkat, sistem akan menambahkan instance baru untuk menampung beban tambahan. Ketika trafik menurun, sistem akan secara otomatis menurunkan jumlah instance untuk menghemat biaya. Sementara itu, *Load Balancer* berperan penting dalam mendistribusikan trafik ke seluruh instance yang aktif secara merata. Dengan fitur health check, *Load Balancer* memastikan bahwa hanya instance yang sehat yang menerima trafik. Kombinasi keduanya menciptakan sistem yang fleksibel, stabil, dan efisien, serta mampu menjaga waktu respon tetap rendah dan ketersediaan layanan tetap tinggi, bahkan dalam kondisi trafik padat sekalipun. Pengujian menunjukkan bahwa sistem dengan *Auto Scaling* dan *Load Balancer* mampu menangani volume trafik yang lebih besar tanpa terjadi penurunan kualitas layanan.

V. KESIMPULAN

Secara keseluruhan, hasil studi kasus Cafe AWS membuktikan bahwa penerapan sistem berbasis *cloud* yang menggabungkan *Auto Scaling* dan *Load Balancer* secara terintegrasi merupakan pendekatan terbaik dalam membangun sistem yang tangguh, efisien, dan responsif. Sistem ini mampu beradaptasi secara otomatis terhadap fluktuasi trafik, meminimalkan risiko kegagalan, mengurangi intervensi manual, serta meningkatkan efisiensi biaya operasional. Sementara sistem tanpa kedua fitur ini terbukti tidak mampu memenuhi kebutuhan performa yang konsisten dalam situasi nyata, pendekatan aktif dengan *Auto Scaling* dan *Load Balancer* justru memungkinkan layanan seperti Cafe AWS untuk beroperasi secara andal, stabil, dan berkelanjutan, sekalipun dalam kondisi trafik yang tinggi dan dinamis. Dengan mempertimbangkan seluruh aspek yang diuji—mulai dari kecepatan respon, kemampuan sistem menghadapi beban tinggi, efisiensi biaya, serta ketersediaan layanan—dapat disimpulkan bahwa kombinasi *Auto Scaling* dan *Load Balancer* merupakan strategi terbaik dalam arsitektur sistem berbasis *cloud* seperti Cafe AWS[17]. Sistem ini memberikan fleksibilitas tinggi, ketahanan terhadap gangguan, dan kemampuan adaptasi otomatis, yang secara langsung mendukung keberhasilan operasional dan kepuasan pelanggan. Dalam konteks aplikasi yang melayani pengguna dalam jumlah besar dan dengan trafik yang fluktuatif, penerapan dua komponen ini tidak hanya menjadi keunggulan teknis, tetapi juga kebutuhan fundamental demi menjaga kualitas layanan secara konsisten dan berkelanjutan.

VI. DAFTAR PUSTAKA

[1] Mell, P., & Grance, T. (2011). The NIST Definition of *Cloud computing*. National Institute of Standards and Technology. Special Publication 800-145. https://doi.org/10.6028/NIST.SP.800-145

[2] Rittinghouse, J. W., & Ransome, J. F. (2017). *Cloud computing*: Implementation, Management, and Security. CRC Press.

[3] Erl, T., Mahmood, Z., & Puttini, R. (2013). *Cloud computing*: Concepts, Technology & Architecture. Prentice Hall.

[4] Amazon Web Services. (2023). About AWS. Retrieved from https://aws.amazon.com/about-aws/

[5] Amazon Web Services. (2023). Amazon EC2 – Virtual *Server*s in the *Cloud*. Retrieved from https://aws.amazon.com/ec2/

[6] Amazon Web Services. (2023). Amazon Virtual Private *Cloud* (VPC). Retrieved from https://aws.amazon.com/vpc/

[7] Gaur, M., & Sharma, A. (2019). Enhancement of Auto Scaling and Load Balancing using AWS.Dalam penelitian ini, AWS digunakan untuk membandingkan performa sistem sebelum dan sesudah penerapan auto scaling dan load balancer. Hasil menunjukkan penurunan response time hingga47%.https://www.researchgate.net/publication/338833518

[8] Chawla, K. (2024). Adaptive Load Balancing in *Cloud computing* Using Reinforcement Learning.Studi ini menggunakan pendekatan pembelajaran penguatan untuk meningkatkan efisiensi load balancing dalam infrastruktur *cloud* dinamis.https://arxiv.org/abs/2409.04896

[9] Jin, Y., & Yang, S. (2025). Scalable Deep Learning-Based Auto Scaling for AI Inference in *Cloud*.Pendekatan deep learning digunakan untuk memprediksi trafik layanan AI dan menyesuaikan kapasitas secara dinamis.https://arxiv.org/abs/2504.15296

[10] Shuja, J., Bilal, K., Hayat, K., Madani, S.A., Khan, S.U. and Shahzad, S. (2012) Energy-Efficient Data Centers. Computing, 94, 973-994.

[11] Heba Nashaat, Nesma Ashry, and Rawya Rizk. 2019. Smart elastic scheduling algorithm for virtual machine migration in *cloud computing*. J. Supercomput. 75, 7 (July 2019), 3842–3865. https://doi.org/10.1007/s11227-019-02748-2

[12] AWS Blog (2022). Scaling Strategies for Elastic Load Balancing.Membahas teknik sharding ELB dan metode penskalaan ALB untuk skenario skala besar di production environment.https://aws.amazon.com/blogs/networking-and-content-delivery/scaling-strategies-for-elastic-load-balancing

[13] Haeruddin, H. (2023, March 20). Ketersediaan Tinggi Infrastruktur Elearning Berbasis Komputasi Awan. *JATISI (Jurnal Teknik Informatika Dan Sistem Informasi)*, *10*(1), 670-682. https://doi.org/https://doi.org/10.35957/jatisi.v10i1.5050

[14] Ramegowda, Ashalatha & Agarkhed, Jayashree. (2015). Evaluation of Auto Scaling and Load Balancing Features in *Cloud*. International Journal of Computer Applications. 117. 30-33. 10.5120/20561-2949.

[15] M. Mangayarkarasi, S. Tamil Selvan, R. Kuppuchamy, S. Shanthi, and S. R. Prem, "*High*ly Scalable and Load Balanced Web *Server* On AWS *Cloud*," IOP Conf. Ser. Mater. Sci. Eng., Vol. 1055, No. 1, p. 012113, 2021, doi: 10.1088/1757-899x/1055/1/012113.

[16] T. Lorido-Botran, J. Miguel-Alonso, and J. A. Lozano, "A Review of Auto-Scaling Techniques for Elastic Applications in *Cloud* Environments," J. Grid Comput., Vol. 12, No. 4, pp. 559–592, 2014, doi: 10.1007/s10723-014-9314-7.

[17] Shahid, M. A., Alam, M. M., & Su'ud, M. M. (2025). A Comprehensive Analysis of Load Balancing in *Cloud computing*: Examining Methodologies and Research Practices for an Effective Hybrid Approach. https://doi.org/10.21203/rs.3.rs-6453751/v1