https://bestijournal.org | PT. Radja Intercontinental Publishing

E-ISSN: 3025-5821

IMPLEMENTATION OF CLOUD COMPUTING ON APPLICATIONS MOBILE JOB INFO

Resha Russita[1], Hari Toha Hidayat[2*], Safriadi[3], Muhammad Nasir[4].
Department of Information and Computer Technology, Politeknik Negeri Lhokseumawe[1,2,3.4]
*Correspondence: haritoha@pnl.ac.id

Abstract
Job vacancy info mobile application has now become an important tool for job seekers in the digital era since the COVID-19 pandemic which recorded a significant spike in the number of unemployed people in Indonesia. In facing this challenge, cloud computing implementation is a crucial solution. This research aims to test performance using the stress testing method and test cloud services from Google Cloud Platform using the Quality of Service (QoS) method. The results show that the server can withstand loads of up to 7000 users simultaneously within 180 seconds or 3 minutes. QoS testing shows that cloud services from Google Cloud Platform achieve an average throughput of 1301.4 kbps with a packet loss rate of 0.54%, an average delay of 0.0019 ms, and an average jitter of 0.00 ms. These results confirm that the implementation of cloud computing in the mobile application of job vacancy information can improve efficiency and quality of service, in accordance with the demands of a dynamic labor market and facing a significant surge in users.

Keywords: *Application, Job Vacancy Info, Cloud, Google Cloud Platform (GCP), Stress Testing, Quality of Service (QoS)*

INTRODUCTION

In today's digital era, mobile applications for job vacancy information have become a key tool for job seekers in finding career opportunities. The COVID-19 pandemic has had a significant impact on the job market, leading to a surge in unemployment in Indonesia. Based on data from the Central Bureau of Statistics (BPS), in 2022, 8.42 million people were unemployed, and despite a decrease in 2023, the number of unemployed people still reached 7.99 million people (BPS - Statistics Indonesia 2022). This situation has further tightened competition among job seekers. In addition, the impact of the pandemic has also forced around 25.6% of companies to lay off workers due to decreased economic activity (Kementerian Koordinator Bidang Perekonomian RI 2021).

Usage of online job search platforms such as JobStreet, LinkedIn, and Indeed has spiked by more than 30%, reflecting a significant shift in people's job search behavior (Jobstreet 2022). The popularity of the gig economy, where individuals are more interested in temporary jobs or flexible projects, has also played a role in this change (Hanivan and Rakhmawan 2023). In the face of these challenges, cloud computing is emerging as an effective technology solution to support mobile job applications. Cloud technology not only provides the scalability required to handle surge in users, but also offers availability high, data security, and cost efficiency.

Google Cloud Platform (GCP) is the main choice in the implementation of cloud computing in this job vacancy mobile application. The products used, namely Compute Engine, and Google Cloud Storage, allow applications to achieve the scale, performance and reliability needed to address complex labor markets. This technology can help workers easily complete profiles and Curriculum Vitae (CV), search for job vacancies with specific filters, save job vacancies, and track the status of their applications. The direct interaction feature between applicants and companies through chat also simplifies the recruitment process.

This research aims to analyze and test the effectiveness of cloud computing implementation on job vacancy mobile application. The main objectives of this research are to measure the application's resilience to high user loads, which is useful in ensuring the quality of user experience on job vacancy mobile applications using the stress testing method, and test the accuracy of cloud performance based on parameters that can be measured, and evaluate the level of success of Google

Cloud Platform services in processing data with certain loads and periods using the Quality of Service method (QoS) with parameters such as Throughput, Packet Loss, Delay, and Jitter.

LITERATURE REVIEW

A. Cloud Computing

Cloud computing is a technology that enables internet- based computing services, where users can access IT resources without needing to understand the infrastructure behind them. By utilizing cloud computing, users can access programs and applications online, including applications designed for Android devices and other connected devices simultaneously. This technology provides the ability to access centralized networks, security, application software, and data storage in an internet environment (Nanda, Hidayat, and Mahlil 2023).

Customizable capacity and traffic is one of the characteristics of cloud computing (Cahya Kurniawan and Amalia 2020). There are three basic cloud computing services, namely IaaS (Infrastructure as a Service), PaaS (Platform as a Service), and SaaS (Software as a Service). This technology provides convenience in terms of scalability and cost efficiency for application development (Wildana 2018).

B. Google Cloud Platform (GCP)

GCP is one of the largest cloud service providers that provides computing infrastructure to store data, run applications, and access various computing resources through the Google cloud network. Services such as Compute Engine, Cloud SQL, and Kubernetes Engine enable the design of infrastructure that is reliable and has a high level of availability (Ramsari and Ginanjar 2022).

C. Laravel

Laravel is a PHP framework designed with MVC (Model- View-Controller) architecture to build fast and efficient web applications. Features such as routing, authentication, and Laravel's template engine make it easier for developers to build robust and integrated web-based applications (Al Hazmi 2018).

D. Flutter

Flutter is a UI framework developed by Google to build mobile applications with high-quality user interfaces. Flutter supports application development on iOS and Android with a single codebase, and has a hot reload feature that allows developers to see code changes in real-time (Saputra and Nudin 2020).

Widgets are the user interface components that build Flutter applications. Each Flutter user interface development will generate a wide variety of widgets tree (Wibowo and Kurniawan 2022). Flutter's output applications can be used on various devices, such as mobile phones, Android and iOS, the web, and even desktops (Wahyu et al. 2023).

E. Quality of Services (QoS)

QoS is a method for measuring network quality of service, including indicators such as throughput, packet loss, delay, and jitter. These parameters are used to evaluate the performance of cloud computing in supporting applications that involve massive user interaction (Hasbi and Saputra 2021).

F. Stress Testing

Stress testing is a method of testing software with the aim of testing the resilience of the application beyond the limits of normal use, especially when there is a surge in users simultaneously, in order to evaluate how the application responds to these extreme conditions (Ginasari, Wibawa, and Wirdiani 2021).

G. JMeter

JMeter is an open-source tool used to perform load testing and stress testing on web applications. JMeter is able to evaluate the strength and stability of applications under heavy load conditions (Ginasari, Wibawa, and Wirdiani 2021).

Performance testing applications such as Jmeter are designed to capture performance at the interface level and can simulate multiple users of an application simultaneously, enabling evaluation of application performance. Applications that can be tailored to this number of users will be very beneficial to organizations because they can describe actual conditions. Applications like this can help companies save time and money (Ade Ismail et al. 2023).

METHOD

The research methodology contains data collection, application architecture design, use case diagrams, and activity diagrams.

A. Data Collection

The data collection methods that the author used in the study are as follows:

1. Literature Study, which is done by collecting material from literatures such as books, articles, and journals related to solving the problems to be studied.
2. Observation, namely the author takes the first step in data collection, namely direct observation of several companies in order to obtain the data needed by the author.
3. Interviews, namely the author asks directly about matters relating to the information needed by the author in the preparation of the final project.
4.

B. Application Architecture Design

This architecture design is useful to ensure that the flow of the system built is clear and achieves the goal of solving storage problems, server damage, and cyberattacks that can disrupt data security. The general design of the system to be implemented in this research can be seen in figure 1.

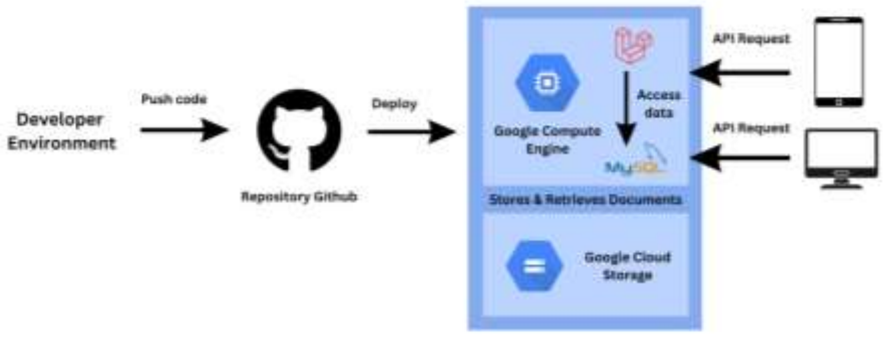


Figure 1. Application architecture design

In figure 1, it is explained that the process starts with the developer writing the code and committing the changes to the Github repository. After that, the code was manually pulled from Github to be deployed to Google Compute Engine (GCE). The application backend is built with the Laravel framework and uses MySQL as the database, while documents and large data are stored on Google Cloud Storage.

Users interact via web and mobile applications, where requests are sent to GCE via Rest API over HTTP/HTTPS protocols. GCE processes the request by first validating the user and authorizing the document. If the validation is successful, the GCE continues the process such as uploading the document to Cloud Storage or retrieving the document from Google Cloud Storage to send back to the user, The response from the GCE is then returned to the user via the web or mobile application, thus ensuring a seamless user experience. This architectural design integrates Github for code management and Google Cloud Platform (GCP) as the cloud infrastructure, enabling a fast and secure deployment process. Rest APIs are used to support consistent and structured interactions between users and the backend, while GCE, Laravel, MySQL, and Google Cloud Storage collaborate to deliver efficient and reliable services.

C. Use Case Diagram

A use case diagram is a visual representation in software engineering that shows the interaction between a system and external actor, such as users or other systems. In the context of a job application, this diagram maps the functionality of the system and assists in design, implementation, and testing by describing the roles of each actor and the relevant use cases.

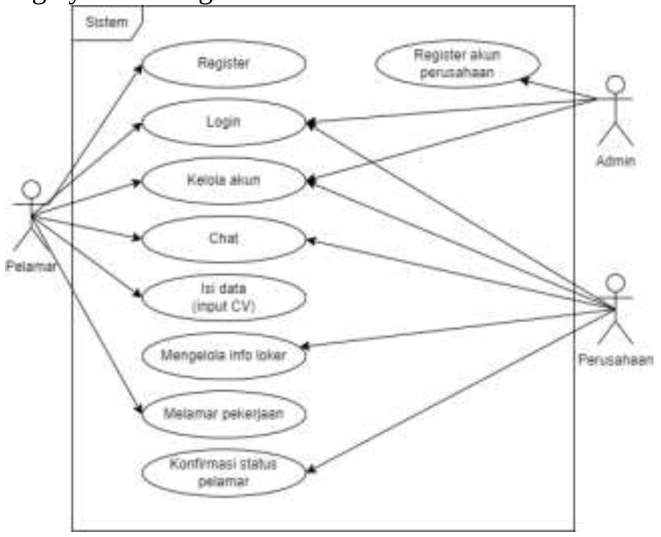


Figure 2. Use case diagram of job vacancy application

In figure 2, a use case diagram is shown which consists of three actors identified as application users mobile job vacancy info. Each actor has a role according to their respective needs. Applicants can apply for jobs, communicate with companies that open vacancies, and register, login, account management, and fill in data in the form of supporting documents to form a CV that is used as a portfolio to apply for a job. On the other hand, companies focus on managing job vacancy information and applicant status, with the account registration process supervised by the admin. Thus, each actor in this job vacancy info mobile application system has a specific role according to their responsibilities, either as an applicant or a company.

D. Activity Diagram

The design of the activity diagram aims to visualize the ongoing business process, namely the process of applicants applying for jobs on job vacancies that have been provided by the company, can be seen in figure 3.

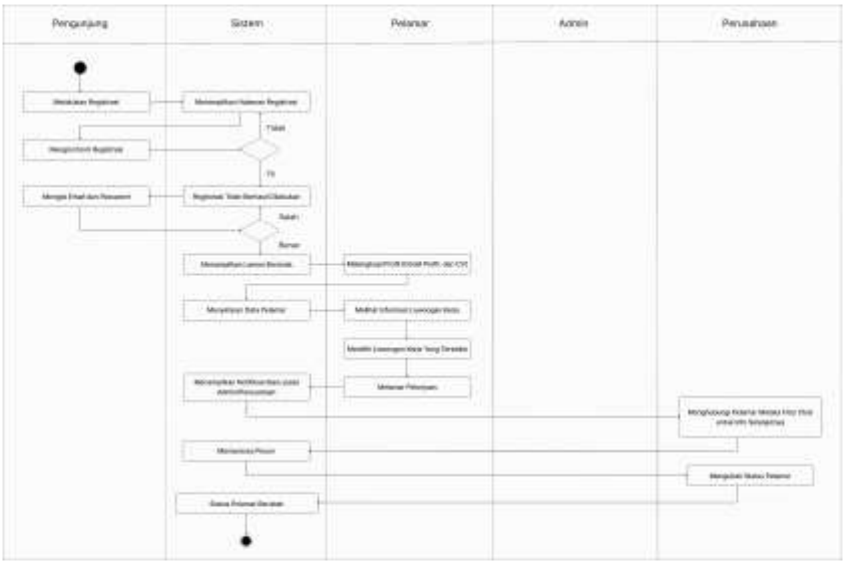


Figure 3. Activity diagram of applying for job vacancies

Upon successful login, the user's status will change to applicant. Applicants complete a profile and CV, which are stored by the system. Applicants can view job vacancies, select and apply for available positions. When an application is submitted, the system sends a notification to the admin and the company. Employers can view the applicant's profile, communicate through the chat feature, and decide whether to accept or reject based on the test, CV, or applicant's profile. The status of the applicant is then changed by the company, and the system saves the changes, placing the accepted applicant in the position applied for.

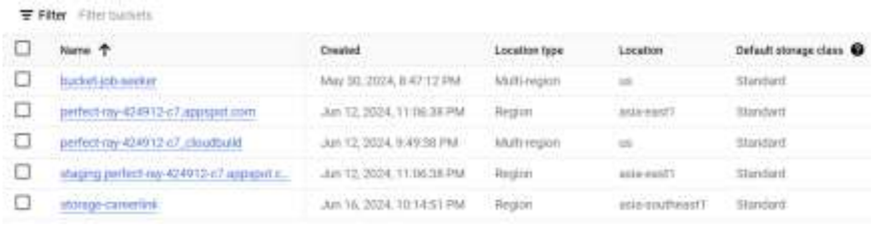
RESULTS AND DISCUSSION

A. Cloud Computing Management on Google Cloud Platform (GCP)

The following is the Cloud Storage interface, and the Compute Engine Instance that has been configured on the GCP.

1. Cloud Storage Bucket Interface

The Cloud Storage interface displays the results of setting up and managing data storage on Cloud Storage. Through this interface, users can manage access permissions, data retention configurations, data withdrawal policies, and other options related to data management. This configuration allows users to optimize data storage according to application needs and security requirements. Thus, data is managed efficiently and securely according to user needs. The Cloud Storage Bucket configuration interface is shown in figure 4.



Name ↑	Created	Location type	Location	Default storage class
bucket-job-seeker	May 30, 2024, 8:47:12 PM	Multi-region	us	Standard
perfect-ray-424912-c7.appspot.com	Jun 12, 2024, 11:06:38 PM	Region	asia-east1	Standard
perfect-ray-424912-c7_cloudbuild	Jun 12, 2024, 9:49:38 PM	Multi-region	us	Standard
staging.perfect-ray-424912-c7.appspot.c...	Jun 12, 2024, 11:06:38 PM	Region	asia-east1	Standard
storage-careerlink	Jun 16, 2024, 10:14:51 PM	Region	asia-southeast1	Standard

Figure 4. Cloud Storage Bucket Interface

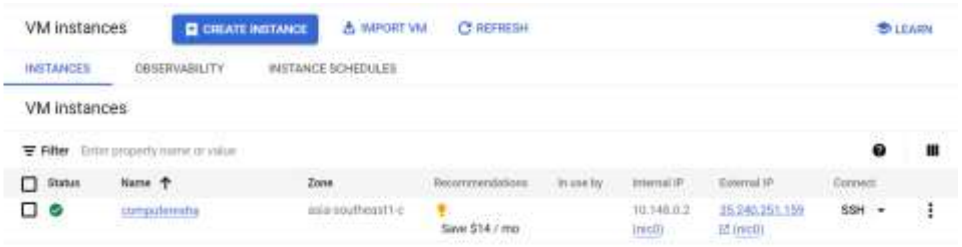
2. Compute Engine Instance Interface

The Instance Coumpute Engine interface on the GCP shows the results of instance settings and configuration including attributes such as CPU, RAM, network, and storage. This interface ensures that the settings match the computing power requirements of the job info mobile application, as described in table I.

TABEL I
JOB VACANCY INFO APPLICATION VM MACHINE SPECIFICATION

Nama	**Nilai**
Machine Type	***e2-small***
Processor	**2 vCPU**
RAM	**2 GB**
Storage	**10 GB HDD**
Zone	***asia-southeast1-c***
OS	**Ubuntu 22.04 LTS**

Users can monitor instance performance in real time, adjust configurations, and optimize resources to support application performance. This helps in managing costs and ensuring applications run smoothly. The Compute Engine instance configuration interface is shown in figure 5.

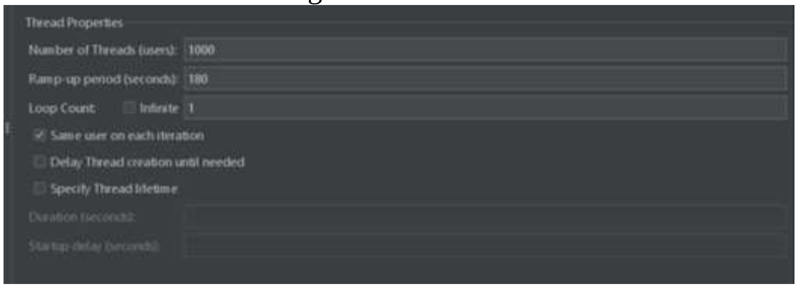


Status	Name	Zone	Recommendations	In use by	Internal IP	External IP	Connect
✓	computeresha	asia-southeast1-c	Save $14 / mo		10.148.0.2 (nic0)	35.240.251.159 (nic0)	SSH

Figure 5. Compute Engine Instance Interface

B. Results of Stress Testing Using JMeter

The following are the results of stress testing using JMeter. This test conducted to evaluate the performance of system stability when facing high loads using JMeter by simulating many users or requests simultaneously by setting the number of request loads (threads), repetition of tests and the time period required. The focus of testing was the http://35.240.251.159/api/jobseeker_list API endpoint used to obtain the applicant list from the application, Several test scenarios were applied with the same variation of request load and ramp up for 180 seconds or 3 minutes. The ramp up was designed to allow the server to adapt to the heavy load as well as ensure consistency within each test scenario. The workload on JMeter is organized through the use of threads that represent virtual users, with each thread running HTTP requests according to predefined scenarios. The test scenario settings on JMeter can be seen in figure 6 below.


Thread Properties

Number of Threads (users): 1000

Ramp-up period (seconds): 180

Loop Count: ☐ Infinite 1

☑ Same user on each iteration

☐ Delay Thread creation until needed

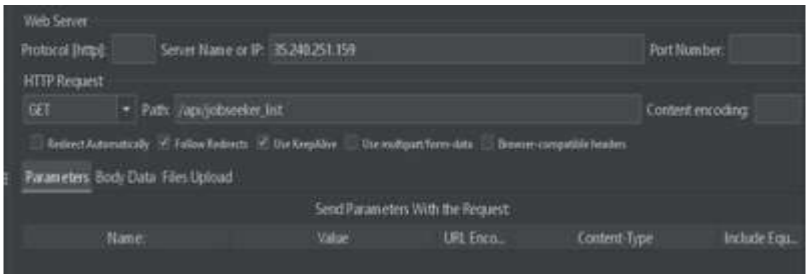
☐ Specify Thread lifetime

Duration (seconds):

Startup delay (seconds):

Figure 6. Test scenario settings on JMeter

Request confirmation is done by specifying the protocol, request method, and relevant endpoint. In this test, the endpoint used is the API http://35.240.251.159/api/jobseeker_list using the HTTP protocol and GET method. This endpoint is used to retrieve the list of observers available in the application. The request settings in JMeter can be seen in figure 7 below.



Web Server
Protocol [http]: Server Name or IP: 35.240.251.159 Port Number:
HTTP Request
GET Path: /api/jobseeker_list Content encoding:
☐ Redirect Automatically ☑ Follow Redirects ☑ Use KeepAlive ☐ Use multipart/form-data ☐ Browser-compatible headers
Parameters Body Data Files Upload
Send Parameters With the Request:

Name:	Value	URL Enco...	Content-Type	Include Equ...

Figure 7. Request settings in JMeter

Stress testing is carried out with various variations of request load (thread), namely 100, 500, 1000, 3000, 5000, 7500, 10000, 15000, 17500, and 20000. Result of the tests are presented in table II below.

TABEL II
SUMMARY OF STRESS TESTING RESULTS

Beban	*Average* (ms)	*Error* (%)	*Throughput* (req/s)
100	122	0	**0.56**
500	114	0	**2.8**
1000	111	0	**5.6**
3000	109	0	**16.7**
5000	117	0.02	**27.8**
7500	7454	4.45	**31.7**
10000	13773	24.15	**41.3**
15000	18821	49	**59.4**
17500	21712	62.7	**63.9**

A comparison of the values in the table above if displayed in graph form can be seen in the following figure.

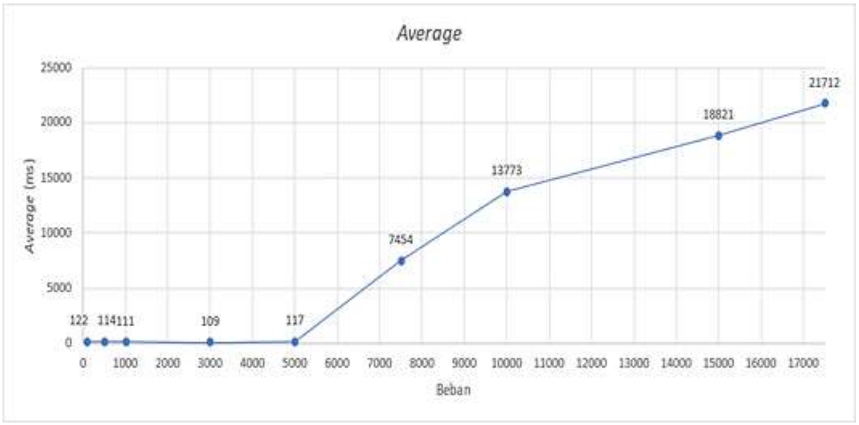


Figure 8. Graph of average parameters in stress testing

Figure 8 displays a graph of the average response time at each load tested. The server response time stabilized below 1 second or 1000 ms at loads 100 to 5000. At a load of 7500, the response time increases from 117 to 7454 ms, indicating a decrease in performance. This increase continues at loads 10000 to 17500, almost double that of the 7500 load. The increasing request load will cause some requests to fail to be processed in other words an error on JMeter.

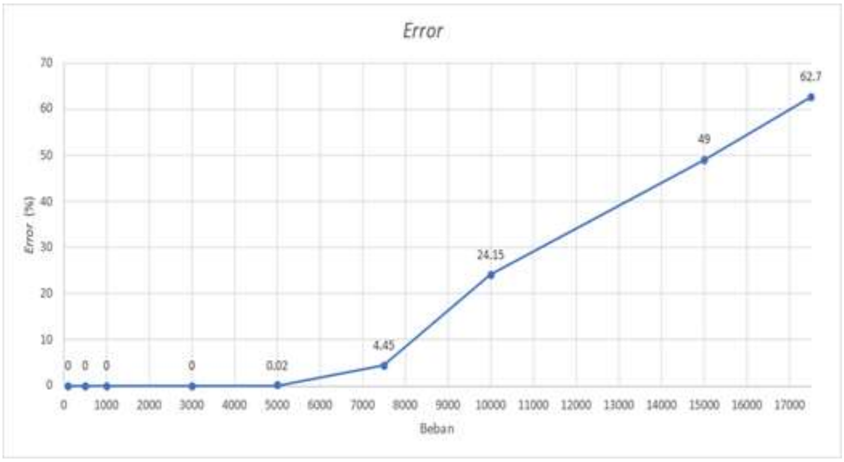


Figure 9. Graph of error parameters in stress testing

Figure 9 shows that the server does not experience errors at loads of 100 to 3000. But at 5000 load, the error reached 0.02% which means that out of 5000 requests, there were requests that failed to be processed. At a load of 7500, the error increases to 4.45% with 7166 out of 7500 requests successfully processed. Increasing load from 10000 to 17500 reached a higher error rate of 24.15% to 62.7%, with only 6527 requests successfully processed. The significant increase in error indicates the server has reached its capacity limit and cannot handle the load within 180 seconds, with a read timeout error indicating the server ran out of time to process the request.

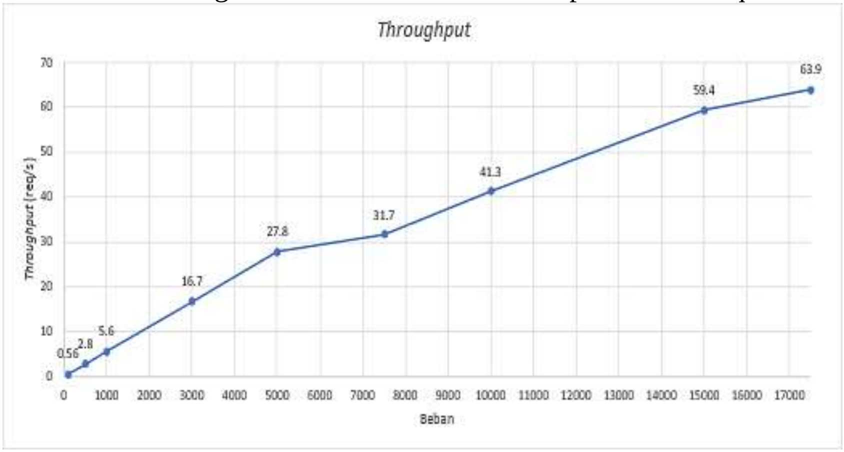


Figure 10. Graph of throughput parameters in stress testing

The graph in figure 10 shows that the throughput increases as the load increases, because the test is done in the same time, which is 180 seconds. At a load of 100 requests, the server processes about 0.56 requests per second. In contrast, at a load of 17500 requests, the server processes up to 65.9 requests per second. Although throughput increased significantly, the server was not always able to process all requests due to performance limits or time constraints.

C. Quality of Services (QoS) Testing Results Using Wireshark

The following are the results of QoS testing after collecting data from 100, 500, 1000, 3000, 5000, 7500, 10000, 15000, and 17500 request loads. This evaluation focused on the performance of the compute engine instance in managing increased network traffic, with measurement parameters including throughput, packet loss, delay, and jitter. The test data was evaluated using TIPHON standardization to determine the quality of service provided. The results of the tests are presented in table III below.

TABEL III
SUMMARY OF QUALITY OF SERVICES (QOS) RESULTS

Beban request	Throughput (ms)	Packet Loss (%)	Delay (ms)	Jitter (ms)
100	906	0	0.01	**0**
500	965	0	0.01	**0**
1000	996	0.1	0.01	**0**
3000	1527	0	0,002	**0**
5000	1248	0	0,002	**0**
7500	1288	1	0.002	**0**
10000	1564	1.4	0.002	**0**
15000	1618	1.2	0.002	**0**
17500	1601	1.2	0.001	**0**
Total	**11713**	**4.9**	**0.037**	**0**
Rata-rata	**1301.4**	**0.54**	**0.0019**	**0**

By comparing all the test values from table III above, we can see the difference in the decrease or increase in values from the graph below.

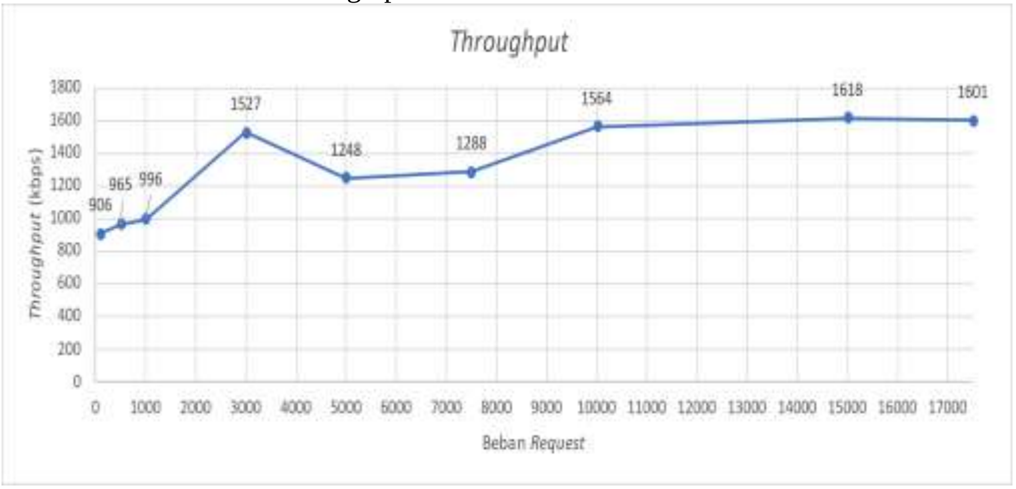


Figure 11. Graph of throughput parameters in QoS testing

Figure 11 shows that the throughput value increases as the number of requests increases, which causes more packets to be captured by Wireshark. The increase in network traffic also contributes to the number of packets captured. The throughput value is affected by the number of requests, observation duration, and network conditions. The instability of throughput at a load of 3000 requests is likely due to network conditions.

The throughput parameter in QoS must have a relationship with packet loss or packets that are lost and do not reach their destination due to errors. A comparison of packet loss parameters can be seen in the following figure.

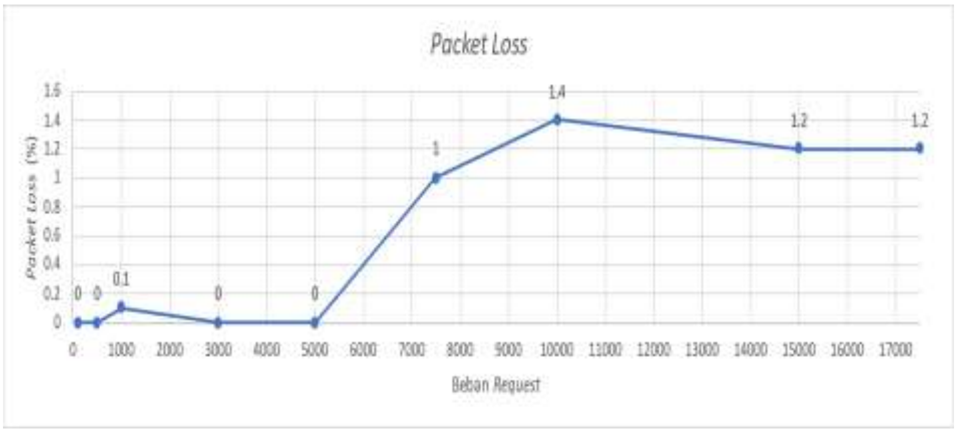


Figure 12. Graph of packet loss parameters in QoS testing

Based on Figure 12, it can be seen that the resulting packet loss value tends to be low and is always below 2%. According to TIPHON standardization, the resulting value is included in the excellent category. This shows that the network used during the observation was of high quality, so errors in packet delivery were minimal.

Another parameter is delay or the time it takes for the packet to reach the receiver. In Figure, the delay value is presented in the form of a graph so that a comparison can be seen based on the variation of the request load.

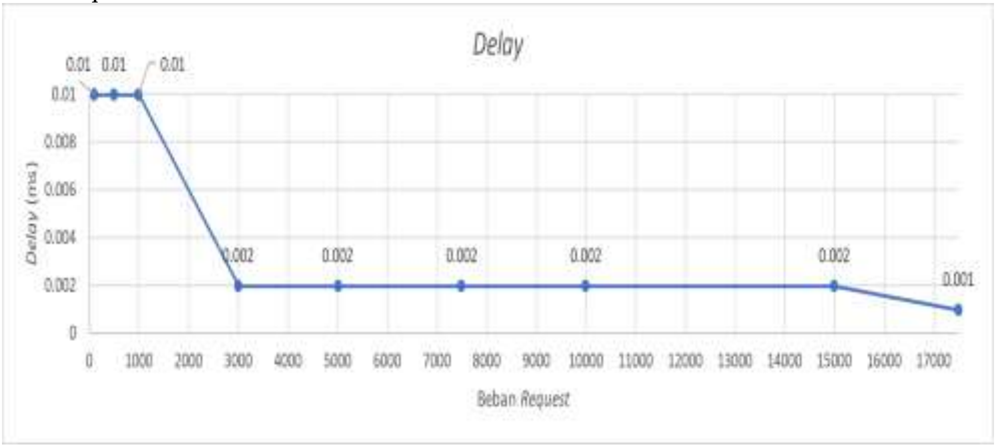


Figure 13. Graph of delay parameters in QoS testing

Figure 13 shows that the resulting delay value tends to decrease as the load of requests sent increases. The resulting delay value is very good according to TIPHON standardization with a value below 150 ms. This decrease in delay indicates that the network is able to handle the packets sent efficiently and reliably. The number of requests that are increased when sent makes the number of packets received also increase and become denser, so the network tries to send all packets with a smaller time difference, ultimately reducing the delay value.

The delay value is greatly affected by jitter, which is the variation in delay. Networks with smaller jitter values show better quality. The results of testing the jitter parameter produce a value of 0 for all variations of request load and are in a very good category according to TIPHON standardization because they are below 75 ms. Although the resulting jitter value is very small, it shows that jitter still exists, but in a very minimal amount. This indicates that the quality of the network used was high during the observation. The total jitter accumulated still shows a very low number. In Figure 28 the total jitter is presented in the form of a graph so that a comparison can be seen based on the variation of the request load.

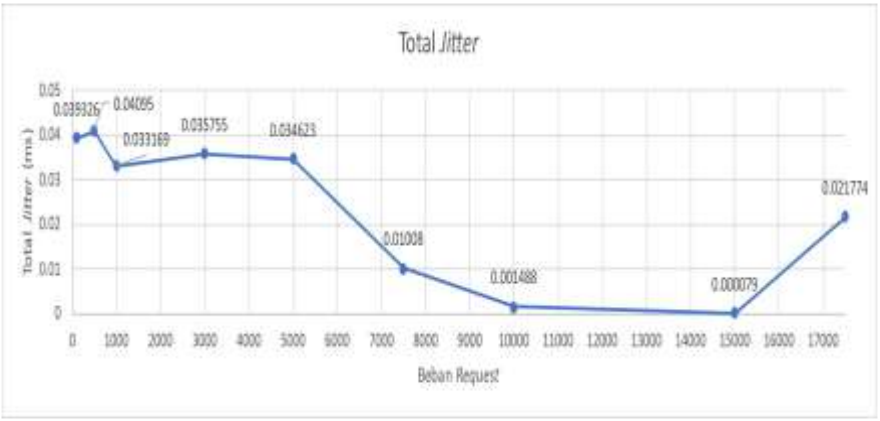


Figure 14. Graph of total jitter parameters in QoS testing

Based on figure 14, it can be seen that the total jitter on the graph looks inconsistent because there is a decrease and increase. This decrease in total jitter indicates that the network quality is getting better at handling the requests sent.

CLOSING

Based on the implementation and testing that has been done on the job vacancy info mobile application system with cloud integration from Google Cloud Platform, it can be concluded that this application is able to handle loads of up to 7000 users within 3 minutes. Stress testing using JMeter shows that when the load is increased to 7500, there is a 4.45% error with only 7166 of the 7500 requests successfully processed by the server. At higher loads of 10000, 15000, and 17500, the error rate increased to 62.7% indicating a significant decrease in server performance. Despite this, QoS testing using Wireshark showed that the application remained efficient in managing data, with an average throughput of 1301.4 kbps and an average packet loss of only 0.54%. Despite an increase in packet loss of up to 1.4% under very heavy request loads, the delay and jitter parameters remained in the excellent category. The average delay reached 0.0019 ms and the average jitter generated was 0 ms. The instability of the parameter values could have been caused by the network connection. The results obtained show that the network can be stable when handling a large and increasing request load based on TIPHON standardization.

REFERENCES

Ade Ismail, Ahmadi Yuli Ananta, Sofyan Noor Arief, and Elok Nur Hamdana. 2023. "Performance Testing Sistem Ujian Online Menggunakan Jmeter Pada Lingkungan Virtual." *Jurnal Informatika Polinema* 9 (2): 159–64. https://doi.org/10.33795/jip.v9i2.1190.

BPS - Statistics Indonesia. 2022. "Tingkat Pengangguran Terbuka (TPT) Sebesar 5,83 Persen Dan Rata-Rata Upah Buruh Sebesar 2,89 Juta Rupiah per Bulan." *Badan Pusat Statistik*.

Cahya Kurniawan, Agung, and Faizatul Amalia. 2020. "Implementasi Teknologi Cloud Computing Untuk E-Learning Berbasis Website Dengan Framework Laravel (Studi Kasus: MAN 9 Jombang)." *Jurnal Pengembangan Teknologi Informasi Dan Ilmu Komputer* 4 (11): 3833–44. http://j-ptiik.ub.ac.id.

Ginasari, Ni Luh Ayu Sonia, Kadek Suar Wibawa, and Ni Kadek Ayu Wirdiani. 2021. "Pengujian Stress Testing API Sistem Pelayanan Dengan Apache JMeter." *Jurnal Ilmiah Teknologi Dan Komputer* 2 (3).

Hanivan, Hilman, and Suryo Adi Rakhmawan. 2023. "Gig Economy During Pandemic in East Java." *East Java Economic Journal* 7 (1): 69–89. https://doi.org/10.53572/ejavec.v7i1.88.

Hasbi, Muhamad, and Naldo Rafli Saputra. 2021. "Analisis Quality of Service (Qos) Jaringan Internet Kantor Pusat King Bukopin Dengan Menggunakan Wireshark." *Universitas Muhammadiyah Jakarta* 12 (1): 1–7. https://jurnal.umj.ac.id/index.php/justit/article/view/13596/7236.

Hazmi, Mohammad Rifqi Al. 2018. “Rancang Bangun Website Mencari Tukang Menggunakan Framework Laravel.” *Jurnal Buana Informatika* 9 (2): 71. https://doi.org/10.24002/jbi.v9i2.1651.

Jobstreet. 2022. “JobStreet Persembahkan #LangsungKerja Bagi Mitra Pencari Dan Penyedia Lowongan Kerja Sebagai Salah Satu Usaha Memudahkan Pencarian Kerja Di Masa Pandemi - JobStreet Indonesia.” 2022. https://id.jobstreet.com/id/career-advice/article/jobstreet-persembahkan-langsungkerja-bagi-mitra-pencari-dan-penyedia-lowongan-kerja-sebagai-salah-satu-usaha-memudahkan-pencarian-kerja-di-masa-pandemi.

Kementerian Koordinator Bidang Perekonomian RI. 2021. “Laporan Kajian Dampak Pandemi Covid-19 Terhadap Ketenagakerjaan Di Indonesia.” Kementerian Koordinator Bidang Perekonomian RI. 2021. https://www.ekon.go.id/source/publikasi/Dampak Pandemi Covid-19 terhadap Ketenagakerjaan Indonesia.pdf.

Nanda, Asri, Hari Toha Hidayat, and Mahlil Mahlil. 2023. “Implementasi Cloud Computing Untuk Media Pembelajaran Interaktif Bahasa Inggris Berbasis Android.” *Journal of Artificial Intelligence and Software Engineering (J-AISE)* 3 (2): 44. https://doi.org/10.30811/jaise.v3i2.4579.

Ramsari, Nopi, and Arif Ginanjar. 2022. “Implementasi Infrastruktur Server Berbasis Cloud Computing Untuk Web Service Berbasis Teknologi Google Cloud Platform.” *Conference SENATIK STT Adisutjipto Yogyakarta* 7. https://doi.org/10.28989/senatik.v7i0.472.

Saputra, Ilham Puji, and Salamun Rohman Nudin. 2020. “Rancang Bangun Aplikasi Siska (Sistem Informasi Karier) Berbasis Android.” *Jurnal Manajemen Informatika* 10 (2): 21–28.

Wahyu, Andrian, Muhammad Affandes, Pizaini Pizaini, Yelfi Vitriani, and Iwan Iskandar. 2023. “Aplikasi E-Commerce Galeri Lembaga Adat Melayu Riau Berbasis Mobile Menggunakan Flutter Menerapkan Metode Waterfall.” *Journal of Information System Research (JOSH)* 4 (2): 458–69. https://doi.org/10.47065/josh.v4i2.2687.

Wibowo, Agung, and Rahadian Kurniawan. 2022. “Pemanfaatan Flutter Pada Fitur Kenaikan Gaji Berkala Dalam Aplikasi Mobile ASN Memayu (Studi Kasus CV. Atsoft Teknologi).” *Automata* 3 (1). https://www.developerlibs.com/2019/12/flutter-lifecycle-.

Wildana, Faiq. 2018. “Implementasi Cloud Computing Di Beberapa Instansi Pemerintahan.” *Masyarakat Telematika Dan Informasi : Jurnal Penelitian Teknologi Informasi Dan Komunikasi* 8 (2): 97. https://doi.org/10.17933/mti.v8i2.105.