PENGARUH INFRASTRUKTUR *HIGH AVAILABILITY* TERHADAP KINERJA DAN RESPONSIVITAS *WEB SERVER* DI GOOGLE CLOUD PLATFORM: ANALISIS PERBANDINGAN

Reza Pratama; Diah Priyawati
Teknik Informatika, Fakultas Komunikasi dan Informatika,
Universitas Muhammadiyah Surakarta

Abstrak

Teknologi *cloud computing* merupakan salah satu teknologi yang muncul akibat perkembangan jaringan internet yang ramai digunakan oleh masyarakat. Permasalahan muncul ketika layanan diakses oleh banyak pelanggan mengakibatkan peningkatan *traffic* data yang signifikan sehingga hal tersebut akan mempengaruhi kinerja *server*. Dalam penelitian ini, penulis mencoba mengatasi permasalahan tersebut menggunakan konsep *High Availability* yaitu *Load Balancing* untuk membagi beban *traffic* antara jalur ke *server* yang satu dengan jalur ke *server* yang lainnya agar tidak terjadi *overload* pada salah satu jalur. Tujuan dari penelitian ini adalah melakukan analisis perbandingan kinerja dan *responsivitas* antara infrastruktur *server* yang menggunakan *high availability* atau *load balancing* dengan *single deployment* tanpa *load balancing* pada lingkungan *cloud Google Cloud Platform* (GCP*)*. Metode yang digunakan dalam penelitian ini adalah metode eksperimental dengan tahapan yang terdiri dari *literature review*, analisis kebutuhan, desain, implementasi, dan pengujian. Hasil penelitian menunjukkan bahwa *server* dengan *load balancing* yang telah diuji dengan Jmeter memiliki kualitas yang lebih baik dari pada *server single*. Pada *server single* memiliki tingkat *error* lebih tinggi dari pada *sever load balancing*. Dengan masing-masing *user* 300;600;900 *server single* memiliki *error* 12%;13,50%;38,5% user maksimal yang digunakan *server single* agar tidak *error* adalah 250 *user*. Sedangkan pada server *load balancing user* maksimal agar tidak mengalami *error* adalah kurang dari 1.000 user. Sehingga *server load balancing* lebih baik dari pada *server single.*

Kata Kunci: Metode Ummi, Pembelajaran Tahsin Al-Qur’an

Abstract

Cloud computing technology is a technology that has emerged due to the development of internet networks which are widely used by the public. Problems arise when services are accessed by many customers resulting in a significant increase in data traffic so that this will affect server performance. In this research, the author tries to overcome this problem using the High Availability concept, namely Load Balancing, to divide the traffic load between the path to one server and the path to another server so that there is no overload on one path. The aim of this research is to conduct a comparative analysis of performance and responsiveness between server infrastructure that uses high availability or load balancing and single deployment without load balancing in the Google Cloud Platform (GCP) cloud environment. The method used in this research is an experimental method with stages consisting of literature review, needs analysis, design, implementation and testing. The research results show that servers with load balancing that have been tested with Jmeter have better quality than single servers. Single servers have a higher error rate than load balancing servers. With 300; 600; 900 users each, a single server has an error of 12%; 13.50%; 38.5%. The maximum user used by a single server to avoid errors is 250 users. Meanwhile, on a load balancing server, the maximum number of users so that there are no errors is less than 1,000 users. So server load balancing is better than a single server.

Keywords: cloud computing, Jmeter, load balancing, web server.

1. PENDAHULUAN

Dunia teknologi informasi setiap waktunya mengalami kemajuan yang semakin canggih sehingga membuat banyak kalangan dengan mudah untuk mengoperasionalkannya dalam bentuk apapun terlebih dengan adanya jaringan internet. Keunggulan penggunaan jaringan internet adalah kecepatan dalam menyampaikan informasi dengan biaya yang terjangkau dan cakupan komunikasi yang luas. Salah satu teknologi yang muncul akibat perkembangan internet dan kini ramai digunakan adalah *cloud computing* atau komputasi awan yang memungkinkan kita menyewa sumber daya teknologi informasi berupa *software*, *processing power*, dan *storage* yang dapat diakses dari manapun melalui internet (’Abidah et al., 2020). Penyewaan *cloud computing* dapat melalui penyedia layanan *cloud* seperti *Google Cloud Platform* (GCP), *Amazon Web Services (AWS)*, *Microsoft Azure*, *IDCloudhost*, *Niagahoster* dan yang lainnya kemudian membayar sesuai dengan kebutuhan yang digunakan atau istilahnya *on-demand* (Rashid & Chaturvedi, 2019). Kelebihan dari penggunaan *cloud computing* adalah dapat menghemat biaya investasi, menghemat waktu, operasional dan manajemen lebih mudah, meningkatkan *availability* serta ketersedian data, dan menghemat biaya operasional pada saat *realibilitas* (Rumetna, 2018).

Manfaat yang ditawarkan oleh *cloud computing* membuat banyak instansi maupun perusahaan pada saat ini mulai berpindah dari penggunaan infrastruktur *on-premise* menjadi *cloud computing* untuk meningkatkan produktivitas dan pelayanan. Layanan pada sebuah perusahaan sudah pasti akan diakses oleh banyak orang untuk mendapatkan informasi sehingga hal tersebut dapat mengakibatkan peningkatan *traffic* data yang signifikan pada *server* (Wijayanti, 2020). Permasalahan muncul apabila beban kerja yang diterima oleh *server* telalu tinggi sehingga dapat mempengaruhi kinerja dari *server* yang digunakan dan yang paling parah dapat terjadi *server down*. Hal tersebut dapat menganggu proses pertukaran data serta kenyamanan dari pengguna layanan sehingga dapat mempengaruhi daya tarik pelanggan organisasi, menurunkan pendapatan dan kehilangan reputasi (Jader et al., 2019).

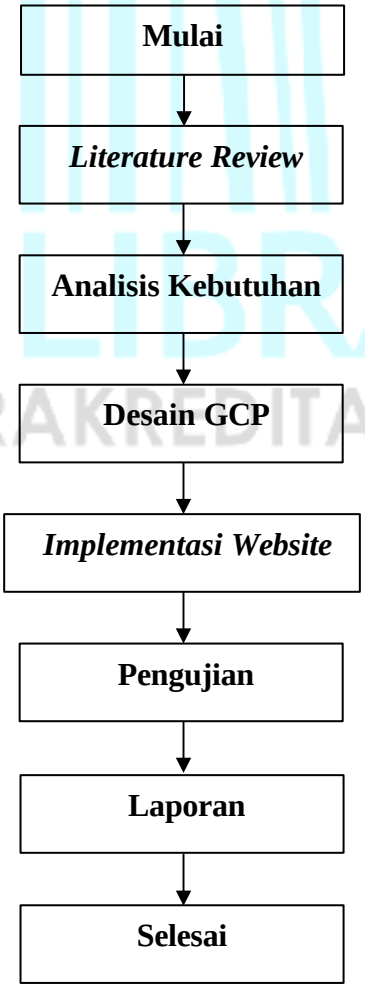
Konsep HA (*High Availability*) merupakan jawaban dari permasalahan tersebut dimana HA merupakan sebuah pendekatan yang digunakan untuk meminimalisir kegagalan pada sebuah sistem atau *server*. Hal tersebut dapat dilakukan dengan adanya *server* duplikasi yang mampu mem-*backup server* agar ketika terdapat gangguan pada *server* maka beban kerja *server* akan digantikan oleh *server* yang lain (Iryani et al., 2022). Salah satu metode HA yang sering digunakan pada *cloud computing* adalah *load balancing*. *Load balancing* digunakan untuk membagi beban *traffic* antara jalur ke *server* yang satu dengan jalur ke *server* yang lainnya agar tidak terjadi *overload* pada salah

satu jalur dan *traffic* dapat berjalan secara optimal sehingga dapat meningkatkan kualitas dan kestabilan dari layanan *server* dengan waktu tanggap yang kecil serta *troughput* yang maksimal (Pratama et al., 2021).

Sehingga dalam penelitian ini penulis bertujuan untuk melakukan analisis perbandingan kinerja dan *responsivitas* antara infrastruktur *cloud web server* yang menggunakan *high availability* atau *load balancing* dengan *single deployment* tanpa *load balancing* pada lingkungan *cloud* GCP. Diharapkan dengan adanya penelitian ini dapat memberikan wawasan yang lebih baik tentang *high availability* terhadap kinerja *web server*, sehingga dapat menjadi panduan bagi administrator sistem dalam mengoptimalkan infrastruktur *web* mereka di lingkungan GCP.

2. METODE

Penelitian ini jenis metode penelitian yang digunakan adalah metode eksperimental yaitu pendekatan yang bertujuan untuk mengetahui perbandingan kelompok eksperimen yang dikenai perlakuan tertentu dengan kelompok kontrol yang tidak dikenai perlakuan (Setyanto, 2006). Adapun diagram alir dari penelitian ini dapat dilihat pada Gambar 1.



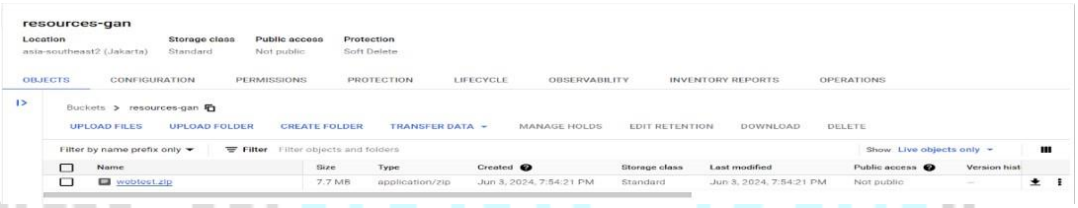
Gambar 1. Diagram Alir Penelitian

3. HASIL DAN PEMBAHASAN

3.1. Hasil Implementasi

3.1.1. Pembuatan Cloud Storage Bucket

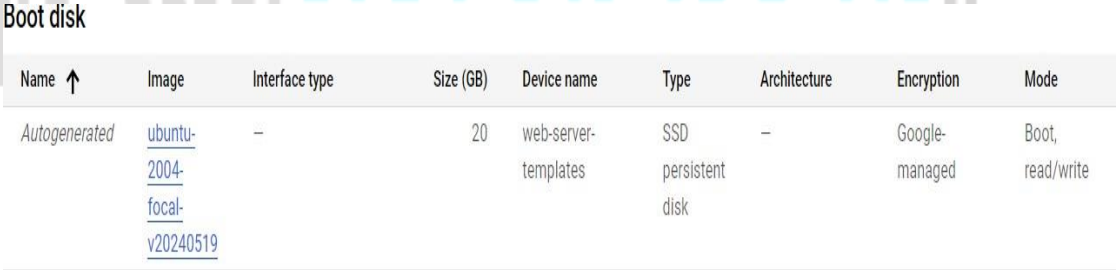
Pembuatan *cloud strorage bucket* bertujuan untuk menyimpan *database website.* Pembuatan nama pada *bucket* digunakan untuk memberikan identitas pada *cloude storage*. Selanjutnya untuk pemilihan lokasi menggunakan satu regional, *website* tersebut akan diakses, dan selebihnya pengaturan *default*. Meng-*upload file website* dalam bentuk zip, dapat dilihat pada gambar 2.



Gambar 2. Website dalam bentuk ZIP

3.1.2. VM Instance Template

Pembuatan VM *Instance* menggunakan *Instance template* agar mempermudah dalam pembuatan *instance single* dan *instance group*. Pemilihan regional disesuaikan dengan *cloud strorage bucket*. *Mechine type* menggunakan standard 2vCPU RAM 8. Selanjutnya, *template instance* dengan penyimanan sebesar 20 GB dengan tipe SSD *persistent disk* sistem operasi ubuntu 20.04 LTS, dapat dilihat pada Gambar 2.



Gambar 3. Sistem Ubuntu

Selanjutnya mengaktifkan semua *firewall* untuk membuka semua *port* agar dapat diakses, dan terakhir pengisian *management automation* yang telah dibuat sebelumnya dapat dilihat pada Gambar 4.

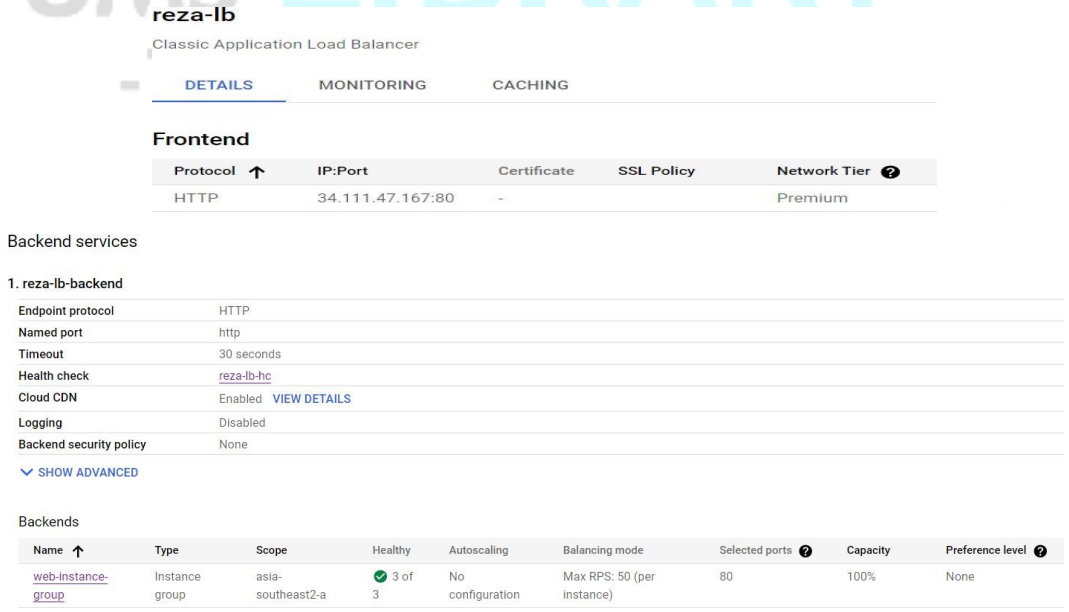
Setelah semua *template instance* berhasil dibuat. Tahap selanjutnya pembuatan VM *Instance single* dengan pengisian otomatis dari *template instance* yang telah dibuat. Setelah VM *instance single* berhasil dibuat akan muncul satu alamat IP. Selanjutnya, pada pembuatan *instance group* dengan menggunakan *tamplate* yang sama dengan VM *single* dengan membuat tiga *server* dengan topologi. Penambahan *health check* dengan *protocol HTTP*.

Key	Value
startup-script	#!/bin/bash # Update package list apt-get update # Install Apache apt-get install -y apache2 # Install UNZIP apt-get install -y unzip # Mengunduh aplikasi web cd /var/www/html/ gsutil cp gs://resources-gan/webtest.zip /var/www/html unzip webtest.zip # Setup aplikasi web chmod -R 777 webtest # Restart Apache sudo systemctl restart apache2

Gambar 4. Pengisian Management Automation

3.1.3. Load Balancing

Tipe *load balancing* yang digunakan adalah *application HTTP/HTTPS*. Menggunakan *public eksternal* sebab permintaan *user* untuk mengakses target menggunakan jaringan internet. Penyebaran yang digunakan adalah penyebaran global agar memudahkan *user* menerapkan *backend* di beberapa wilayah dengan menggunakan IP. *Load balancing* generation menggunakan *classic application* hal tersebut digunakan karena *load balancing* generasi *global* mendukung manajemen lalu lintas tingkat lanjut. Langkah selanjutnya membuat layanan *backend*. Pembuatan *health check* pada *backend load balancing*. Setelah pembuatan *load balancing* berhasil akan mendapatan satu alamat IP. Ketika *user* mengakses IP yang dihasilan *load balancing* maka akan diarahkan pada salah satu IP *server* yang dihasilkan dari VM *instance group*. Dapat dilihat pada gambar 5.



Gambar 5. Hasil IP Load Balancing

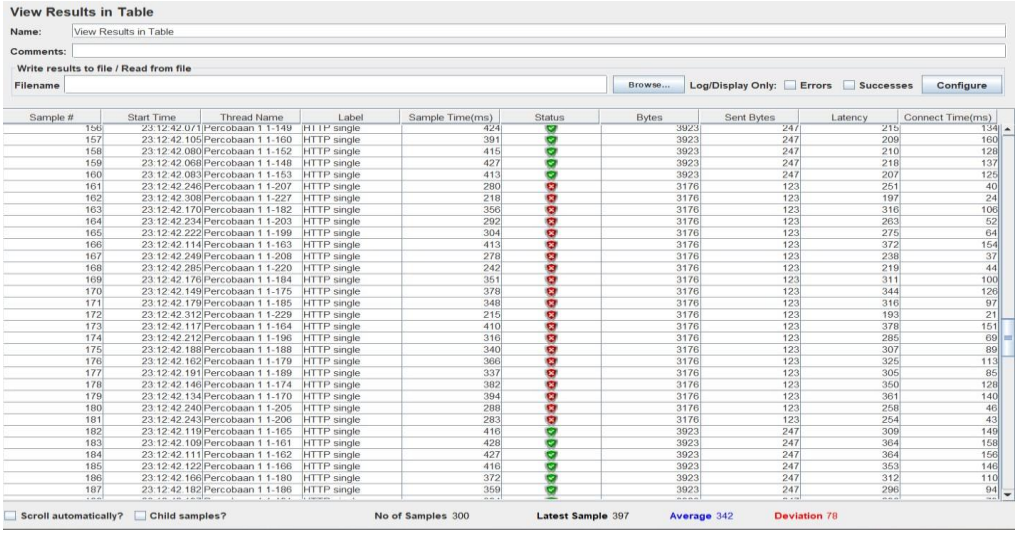
3.2. Hasil Penelitian

Pengujian yang dilakukan dalam penelitian ini adalah pengujian dengan menggunakan JMeter. Jmeter merupakan salah satu aplikasi yang masih sangat popular. Selain itu, Jmeter digunakan karena jmeter dirancang untuk memuat tes perilaku fungsional dan mengukur kinerja program tersebut. Jmeter dirancang untuk dapat digunakan untuk pengujian kinerja pada *website.* Pengujian yang terdistribusikan tersebut menjadi salah satu kelebihan dari Jmeter. Selain itu, banyak pilihan fitur yang cukup lengkap pada jmeter. Penggunaan Jmeter harus di *support* dengan adanya *javascript*. Tujuan adanya *javascript* adalah untuk membaca pemrograman *website* (Alam & Dewi, 2022).

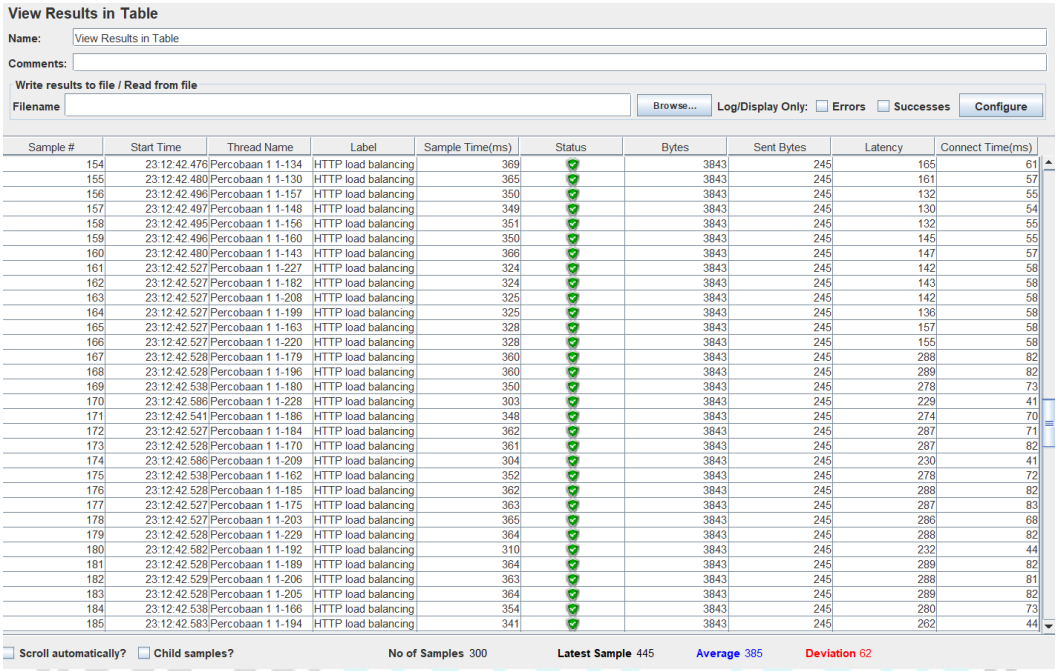
Pengujian *server* dengan Aplikasi Jmeter dilakukan menggunakan tiga percobaan. Pengujian dilakukan dengan bertahap dan bervariasi menggunakan teknik peningkatan jumlah yang dikenal dengan teknik "*step load"* yaitu dengan masing-masing percobaan ditambahkan 250 *user* dari jumlah percobaan sebelumnya dengan *rum-up periode (second)* 1 dan *loop count* 1 pada masing-masing IP *single* maupun *load balancing*. Hal ini akan efektif untuk mengevaluasi kinerja sistem dengan berbagai kondisi pengguna.

Pengujian *server* menggunakan *View result in table dan Summary Report*. *View result in table* digunakan untuk menampilkan hasil pengujian dalam format *table* sedangkan *Summary report* digunakan untuk menyajikan ringkasan dari hasil pengujian. Gambar 6 dan gambar 14 menunjukkan data yang diperoleh dari *View Result InTable.*

Hasil pengujian dengan *view result in table* pada *server single* dan *load balancing* pada Gambar 6 dan Gambar 14 memiliki perbedaan. *Server single* bersatatus terdapat *user* yang dapat mengakses dan terdapat *user* yang tidak dapat mengakses. Sedangkan pada *server load balancing user* memiliki status dapat diakses semua. Data selanjutnya menggunakan *Summary Report.*



Gambar 6. View In Table Server Single



View Results in Table

Name: View Results in Table

Comments:

Write results to file / Read from file

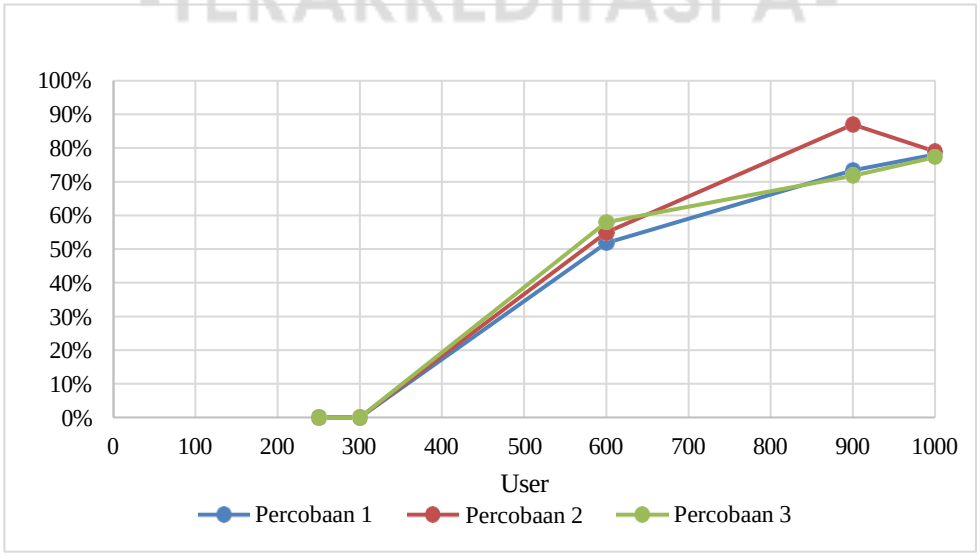
Filename | Browse... | Log/Display Only: Errors Successes | Configure

Sample #	Start Time	Thread Name	Label	Sample Time(ms)	Status	Bytes	Sent Bytes	Latency	Connect Time(ms)
154	23:12:42.476	Percobaan 1 1-134	HTTP load balancing	369		3843	245	165	61
155	23:12:42.480	Percobaan 1 1-130	HTTP load balancing	365		3843	245	161	57
156	23:12:42.496	Percobaan 1 1-157	HTTP load balancing	350		3843	245	132	55
157	23:12:42.497	Percobaan 1 1-148	HTTP load balancing	349		3843	245	130	54
158	23:12:42.495	Percobaan 1 1-156	HTTP load balancing	351		3843	245	132	55
159	23:12:42.496	Percobaan 1 1-160	HTTP load balancing	350		3843	245	145	55
160	23:12:42.480	Percobaan 1 1-143	HTTP load balancing	366		3843	245	147	57
161	23:12:42.527	Percobaan 1 1-227	HTTP load balancing	324		3843	245	142	58
162	23:12:42.527	Percobaan 1 1-182	HTTP load balancing	324		3843	245	143	58
163	23:12:42.527	Percobaan 1 1-208	HTTP load balancing	325		3843	245	142	58
164	23:12:42.527	Percobaan 1 1-199	HTTP load balancing	325		3843	245	136	58
165	23:12:42.527	Percobaan 1 1-163	HTTP load balancing	328		3843	245	157	58
166	23:12:42.527	Percobaan 1 1-220	HTTP load balancing	328		3843	245	155	58
167	23:12:42.528	Percobaan 1 1-179	HTTP load balancing	360		3843	245	288	82
168	23:12:42.528	Percobaan 1 1-196	HTTP load balancing	360		3843	245	289	82
169	23:12:42.538	Percobaan 1 1-180	HTTP load balancing	350		3843	245	278	73
170	23:12:42.586	Percobaan 1 1-228	HTTP load balancing	303		3843	245	229	41
171	23:12:42.541	Percobaan 1 1-186	HTTP load balancing	348		3843	245	274	70
172	23:12:42.527	Percobaan 1 1-184	HTTP load balancing	362		3843	245	287	71
173	23:12:42.528	Percobaan 1 1-170	HTTP load balancing	361		3843	245	287	82
174	23:12:42.586	Percobaan 1 1-209	HTTP load balancing	304		3843	245	230	41
175	23:12:42.538	Percobaan 1 1-162	HTTP load balancing	352		3843	245	278	72
176	23:12:42.528	Percobaan 1 1-185	HTTP load balancing	362		3843	245	288	82
177	23:12:42.527	Percobaan 1 1-175	HTTP load balancing	363		3843	245	287	83
178	23:12:42.527	Percobaan 1 1-203	HTTP load balancing	365		3843	245	286	68
179	23:12:42.528	Percobaan 1 1-229	HTTP load balancing	364		3843	245	288	82
180	23:12:42.582	Percobaan 1 1-192	HTTP load balancing	310		3843	245	232	44
181	23:12:42.528	Percobaan 1 1-189	HTTP load balancing	364		3843	245	289	82
182	23:12:42.529	Percobaan 1 1-206	HTTP load balancing	363		3843	245	288	81
183	23:12:42.528	Percobaan 1 1-205	HTTP load balancing	364		3843	245	289	82
184	23:12:42.538	Percobaan 1 1-166	HTTP load balancing	354		3843	245	280	73
185	23:12:42.583	Percobaan 1 1-194	HTTP load balancing	341		3843	245	262	44

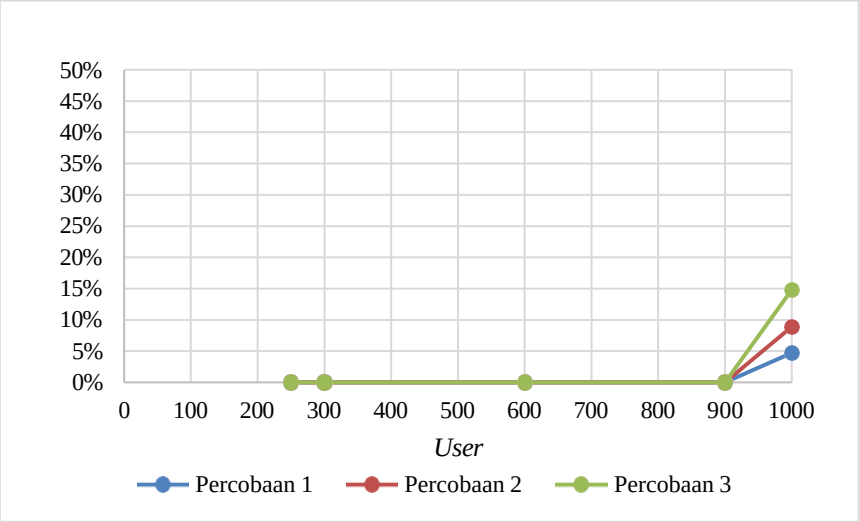
Scroll automatically? Child samples? No of Samples 300 Latest Sample 445 Average 385 Deviation 62

Gambar 7. View In Table Server Load Balance

Pengujian yang dilakukan pada server *Single* maupun *Server load Balancing* mendapatkan hasil rata-rata *request*, *error*, maupun *throughput* yang berbeda. Pengujian *error* untuk mengetahui rasio atau perbandingan antara respon yang benar dan respon yang gagal selama periode tertentu. Hasilnya dihitung dalam bentuk persentase (%). Semakin kecil *error rate* semakin bagus sistem. Sedangkan pengujian *throughput* mengacu pada volume rata-rata data yang benar-benar dapat melintasi jaringan selama waktu tertentu *throughput* menunjukkan jumlah paket data yang berhasil tiba di tujuan dan kehilangan paket data.

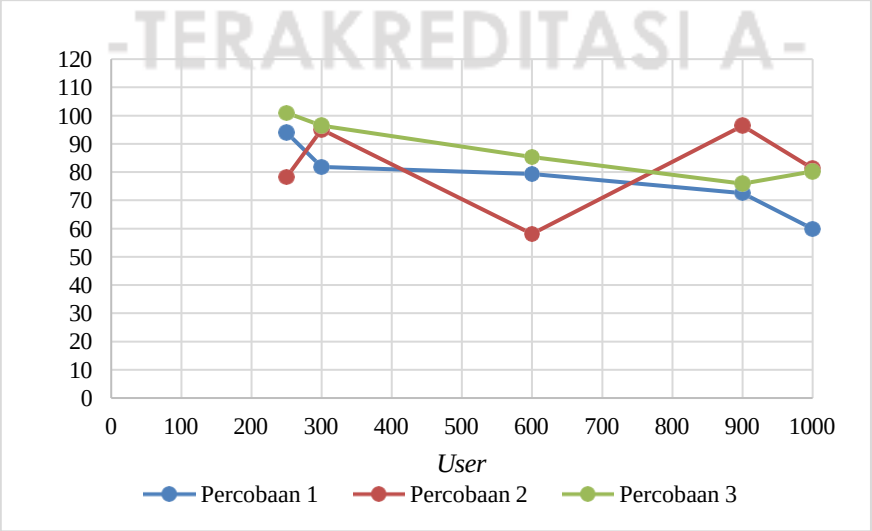


Gambar 8. Pengujian Error Pada Server Single



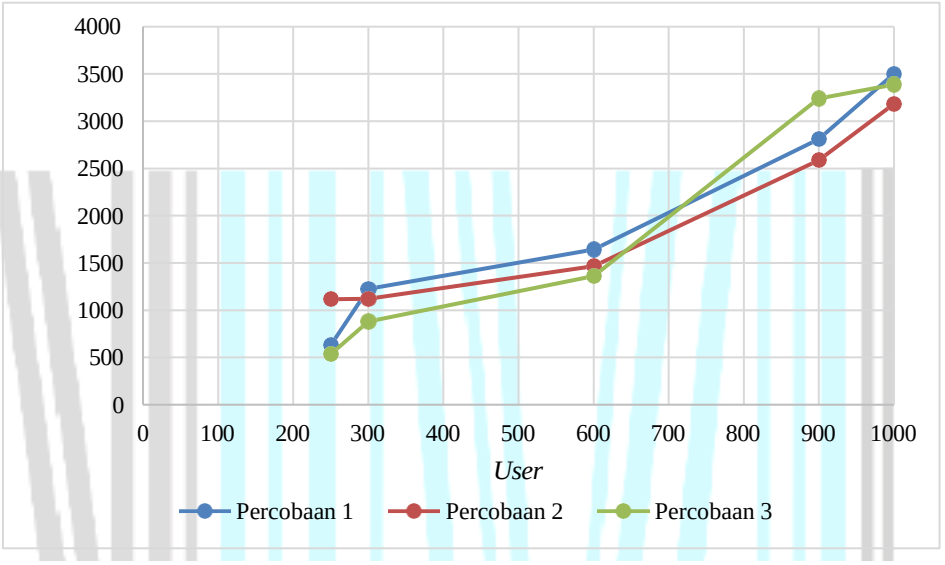
Gambar 9. Pengujian Error Pada Server Load Balancing

Gambar 8 . Pengujian *Error* Pada *Server Load Balancing* Gambar 8. Menunjukkan hasil pengujian *error* pada *server single* pada user 250 dan 300 mendapatkan hasil 0% atau tidak terdeteksi *error*. Percobaan 1 sampai 3 pada *user* 600 sudah mulai mengalami *error* dengan kisaran angka 50%. Selanjutnya hingga *user* 1000 tingkat *error server single* semakin tinggi. Sedangkan pada Gambar 7. Menunjukkan grafik dari hasil pengujian *error* pada *server load Balancing*, pada pengujian 1 sampai pengujian 3 *user* 250-900 mendapatkan hasil *error* 0% yang artinya tidak terdeteksi *error*. Sedangkan mulai terlihat pada *user* 1000 dari pengujian pertama hingga ketiga terus mengalami peningkatan. Hal tersebut terjadi karena semakin banyak *user* yang melakukan permintaan pada *server* sehingga *error* yang dihasilkan semakin tinggi.

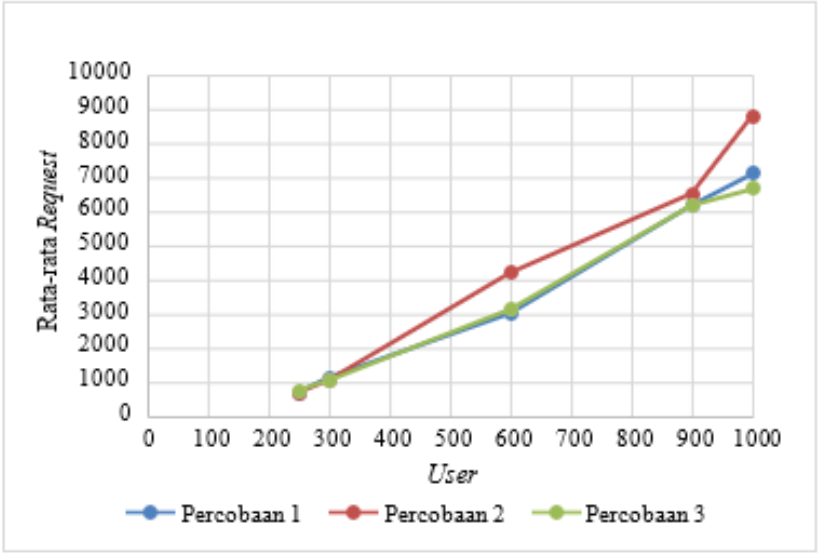


Gambar 10. Pengujian Throughput pada Server Single

Selanjutnya dapat dilihat pada Gambar 10. Terlihat grafik hasil pengujian *Throughput* pada *Server Single*, maupun grafik pada Gambar 11, pada pengujian *Throughput server load balanching* menghasilkan garis grafik yang tidak menentu, hal tersebut disebabkan karena kecepatan pada *throughput* dipengaruhi oleh, *bandwidth*, daya pemrosesan, kehilangan paket dan juga topologi jaringan. Sehingga *throughput* yang dihasilkan pada masing-masing *server* tidak menentu. Pengujian *throughput* yang dihasilkan tidak berpengaruh pada pengujian server yang dilakukan, karena tinggi rendahnya data yang di hasilkan dari *througphput* di pengaruhi oleh *bandwidth* jaringan pada setiap waktunya.



Gambar 12. Rata-rata Request pada Server Single



Gambar 11. Rata-rata Request Server Load Balancing

Gambar 11. Dapat dilihat bahwa rata-rata *request* pada *server single* mengalami peningkatan. Semakin banyak *user* yang akan mengakses semakin tinggi juga rata-rata *request* yang dihasilkan. Selain itu, pada Gambar 12. dapat dilihat bahwa,rata-rata *request* yang

dihasilkan dari *server load balancing* juga mengalami peningkatan pada percobaan pertama dan ketiga didapatkan rata-rata *request* yang dihasilkan hampir sama. Sama seperti *sever single*, rata-rata *request* pada *server load balancing* juga, akan mengalami peningkatan apabila jumlah *user* semakin banyak. Hal yang mempengaruhi perbedaan jumlah rata-rata *request* pada setiap percobaan adalah kondisi lingkungan, perubahan parameter, pengaruh sumber daya berupa server atau koneksi jaringan yang digunakan dapat mempengaruhi hasil permintaan yang berpengaruh pada rata-rata *request.*

4. PENUTUP

Berdasarkan hasil analisis yang telah dilakukan, dapat disimpulkan bahwa penulis mengangkat sebuah topik *cluod computing* dengan menggunakan layanan GCP serta mengembangkan inovasi dengan menggunakan layanaan *load balancing* pada GCP tersebut. Penelitian ini dilakukan pengujian pada *server single* dan *load balancing* untuk membandingkan dan membuktikan bahwa *server load balancing* lebih *efisien* dan lebih baik dari pada *server single*. Dibuktikan dengan pengujian Jmeter didapatkan hasil:

a) Berdasakan tingkat *error* pada *server*, *server single* dengan jumlah *user* maksimal yaitu 250 tidak mengalami *error*, sedangakan pada jumlah *user* mulai dari 300 sudah mengalami *error*. Berbeda dengan *server load balancing* dengan *user* kurang dari 1000 tidak mengalami *error*.
b) Pengujian *throughput* yang dihasilkan tidak berpengaruh pada pengujian *server* yang dilakukan, karena tinggi rendahnya data yang di hasilkan dari *througphput* di pengaruhi oleh *bandwidth* jaringan pada setiap waktunya.
c) Rata-rata *request* yang di hasilkan dari *server single* dan *load balancing*, semakin banyak jumlah *user* maka akan menghasilkan jumlah rata-rata *request* semakin tinggi.

Sehingga dapat dilihat bahwa *server load balancing* lebih baik dari pada *server single.* Diharapkan pada pengembangan penelitian berikutnya untuk lebih memanfaatkan fitur-fitur pada *google clound computing*. Selain itu, peneliti selanjutnya dapat menggunakan *web* dinamis dengan pengujian yang lebih kompleks.

DAFTAR PUSTAKA

’Abidah, I. N., Hamdani, M. A., & Amrozi, Y. (2020). Implementasi Sistem Basis Data Cloud Computing pada Sektor Pendidikan. Keluwih: Jurnal Sains Dan Teknologi, 1(2), 77–84. https://doi.org/10.24123/saintek.v1i2.2868

Alam, E. N., & Dewi, F. (2022). Performance Testing Analysis Of. 06(02), 146–155.

Dirgantara, U., & Suryadarma, M. (2014). Rancang Bangun Penerapan Model Prototype Dalam

Perancangan Sistem Informasi Pencatatan Persediaan Barang Berbasis Web. Jurnal Sistem Informasi Universitas Suryadarma, 8(2), 223–230. https://doi.org/10.35968/jsi.v8i2.737

Girish, Ali, D. A., Samarth, Shrinidhi, & B, N. (2023). Load Testing. International Research Journal of Modernization in Engineering Technology and Science, 05(07), 506–514. https://doi.org/10.1016/b978-075065077-9/50006-x

Google Cloud Platform. (2023a). Cloud Load Balancing overview | Google Cloud. Google Cloud Platform. (2023b). Product overview of Cloud Storage | Google Cloud.

Google Cloud Platform. (2023c). Virtual machine instances | Compute Engine Documentation | Google Cloud.

Iryani, N., Ayatri, K. D., & Wahyuningrum, R. D. (2022). Analisis performansi high availability cluster server menggunakan heartbeat pada private cloud. JITEL (Jurnal Ilmiah Telekomunikasi, Elektronika, Dan Listrik Tenaga), 2(2), 129–138. https://doi.org/10.35313/jitel.v2.i2.2022.129-138

Jader, O. H., Zeebare, S. R. M., & Zebari, R. R. (2019). A state of art survey for web server performance measurement and load balancing mechanisms. International Journal of Scientific and Technology Research, 8(12), 535–543.

Pratama, K. A., Subagio, R. T., Hatta, M., & Asih, V. (2021). Implementasi Laod Balancing Pada Web Server Menggunakan Apache Dengan Server Mirror Data Secara Real Time. Jurnal Digit, 11(2), 178. https://doi.org/10.51920/jd.v11i2.203.

Rashid, A., & Chaturvedi, A. (2019). Cloud Computing Characteristics and Services A Brief Review. International Journal of Computer Sciences and Engineering, 7(2), 421–426. https://doi.org/10.26438/ijcse/v7i2.421426

Rumetna, M. S. (2018). Pemanfaatan Cloud Computing pada Dunia Bisnis: Studi Literatur. Jurnal Teknologi Informasi Dan Ilmu Komputer, 5(3), 305–314. https://doi.org/10.25126/jtiik.201853595

Samrit, & Monika. (2023). A Pragmatic Evaluation of Performance Testing for E-commerce Web based Applications. Journal of Emerging Technologies and Innovative Research (JETIR), 10(8), 718–745.

Setyanto, A. E. (2006). Memperkenalkan Kembali Metode Eksperimen dalam Kajian Komunikasi.

Jurnal ILMU KOMUNIKASI, 3(1). https://doi.org/10.24002/JIK.V3I1.239

Wijayanti, W. (2020). High Performance Database Server (High Availability Database Server) Menggunakan Mariadb Galera Cluster. Universitas Muhammadiyah Surakarta, 1–15.

Yuliadi, Rodianto, Rusdan, & Sofyan MZ, D. (2020). Sistem Informasi Layanan Administrasi Kepegawain Berbasis Lokal Area Network (LAN). Jurnal Informatika, Teknologi Dan Sains, 2(4), 256–259. https://doi.org/10.51401/jinteks.v2i4.829