# Studi Komparasi Performa NGINX dan HAPROXY Sebagai Load Balancer di Cloud Menggunakan Teknologi Kontainer

Joshua Gibeon Mulyana[1], Willy Sudiarto Raharjo[2], Gani Indriyanta[3]
*Program Studi Informatika, Universitas Kristen Duta Wacana Yogyakarta*
joshua.gibeon@ti.ukdw.ac.id
willysr@staff.ukdw.ac.id
ganind@staf.ukdw.ac.id

***Abstract*— High availability and scalability is a must have feature for server that owned by industrial sector. One concept that used by technology practitioner is horizontal scaling. Horizontal scaling can be achieved by using load balancer. Even though those technology practitioner started to adopt containerization technology, those technology practitioner still using load balancer. The purpose of this research is to compare NGINX and HAPROXY performance as load balancer based on response time and error rate. Both NGINX and HAPROXY will run on docker that installed on various virtual machine type in both AWS and GCP. The result shows that based on writer's configuration, NGINX could handle medium and heavy load better than HAPROXY. Another result shows that AWS could handle medium and heavy load better than GCP.***

**Intisari— High availability dan scalability sudah menjadi suatu keharusan bagi server yang dimiliki oleh sektor industri. Salah satu konsep yang dipakai oleh pelaku teknologi adalah horizontal scaling. Horizontal scaling ini dapat dicapai oleh load balancer. Meskipun para pelaku teknologi mulai mengadaptasi teknologi kontainerisasi, para pelaku teknologi ini tetap memakai load balancer. Penelitian ini bertujuan untuk membandingkan performa NGINX dan HAPROXY sebagai load balancer dari sisi response time dan error rate, yang dijalankan diatas docker yang berada di virtual machine AWS dan GCP. Perbandingan ini dilakukan di berbagai spesifikasi virtual machine. Hasil penelitian menunjukkan bahwa berdasarkan konfigurasi yang sudah dibuat olehpenulis, NGINX dapat menangani beban sedang dan besar dengan lebih baik dibandingkan dengan HAPROXY. Hasil penelitian juga menunjukkan AWS dapat menangani beban sedang dan besar dengan lebih baik dibandingkan dengan GCP.**

***Kata Kunci*— Load balancer, Cloud Computing,Docker, Container, Load Testing.**

## I. PENDAHULUAN

Kontainerisasi adalah sebuah teknologi ringan dan portabel yang mempunyai kemampuan untuk mengembangkan, melakukan uji coba, dan melakukan *deployment* terhadap sebuah aplikasi ke server dalam jumlah besar, serta kemampuan untuk terhubung antara satu kontainer dengan kontainer yang lain [1]. Karena kemampuannya tersebut, sudah banyak pelaku teknologi yang mulai berpindah dari teknologi VM (*virtual machine)* ke teknologi kontainerisasi ini. Menurut data yang diperoleh Diamanti [2], muncul pertanda bahwa teknologi kontainerisasi sudah mulai menjadi teknologi *mainstream,* 44% responden berencana untuk mengganti VM dengan kontainer. Walaupun adanya perubahan penggunaan teknologi dari teknologi VM menjadi teknologi kontainerisasi, beberapa prinsip yang digunakan masih sama.

High availability dan scalability menjadi suatu keharusan bagi server yang dimiliki oleh beberapa sektor industri seperti perbankan. Salah satu konsep yang dipakai oleh pelaku teknologi adalah horizontal scalability. Horizontal scalability menurut Anandhi [3] adalah penambahan beberapa unit resource ke dalam sistem dan memperlakukannya sebagai satu kesatuan. Untuk memperlakukannya sebagai satu kesatuan dibutuhkan sesuatu agar semua unit tersebut dapat diperlakukan sebagai satu sistem yang sama.

Load balancer merupakan sesuatu yang dibutuhkan untuk memperlakukan beberapa unit resource tersebut sebagai satukesatuan, load balancer biasanya berada di antara client dan kumpulan server [4]. load balancer bekerja dengan cara membagi traffic atau permintaan yang masuk ke sebuah kumpulan server. Ada beberapa alat yang dapat digunakan untuk melakukan load balancing, beberapa diantaranya adalah NGINX dan HAPROXY.

Penelitian tentang *load balancer* pernah dilakukan oleh Sampurna [5], Sampurna melakukan penelitian yang bertujuan untuk mengevaluasi metode *load balancing* dan *fault tolerance* pada server *chatting* di aplikasi media sosial. Sampurna mencoba membandingkan algoritma *round robin* dan *least connection*. Dari pengujian tersebut didapatkankesimpulan bahwa *least connection* lebih unggul dibanding algoritma *round robin*. Kesimpulan ini dibuktikan dengan *response time* yang lebih kecil dan *throughput* yang lebih besar.

Pada tahun berikutnya, penelitian implementasi *load balancing* pada server basis data di kontainer dilakukan olehMoch. Wahyu Imam Santosa dan kawan kawan [6]. Penelitian dimaksudkan untuk menyelesaikan permasalahan distribusi beban server basis data. Wakyu

menggunakan *load balancer* untuk mendistribusikan bebannya, dan mengukur *throughput* yang didapatkan. Dari hasil pengujian, didapatkan kesimpulan bahwa kinerja server basis data meningkat. Setelah penelitian tentangimplementasi *load balancing* di kontainer yang dilkaukan oleh Wahyu Imam Santosa. Penelitian perbandingan performa *load balancer* diantara kontainer dan *native* dilakukan oleh Fatar Nur Alam Majid [7]. Fatar Nur Alam Majid menguji perbandingan performa implementasi aplikasi *load balancer* yang digunakan secara *native* dengan*load balancer* yang memanfaatkan teknologi kontainer. Hasil penelitian yang dilakukan oleh Fatar menunjukkan bahwa *load balancer* yang dipasang secara *native* dengan *load balancer* yang memanfaatkan teknologi kontainer mempunyai performa yang hampir sama dengan tingkat keyakinan sebesar 99% jika dilihat dari aspek tingkat *availability* dan *response time*.

Dengan mulainya teknologi kontainerisasi menjadi teknologi mainstream saat ini, dengan kebutuhan *high availability* dan *scalability*. Penulis akan membandingkan performa dari NGINX dan HAPROXY sebagai *load balancer*, dilihat dari sisi *error rate* dan *response time*.

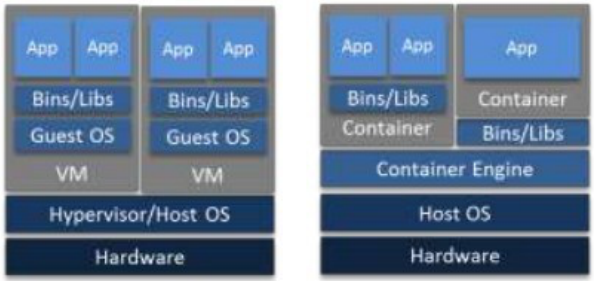
## II. LANDASAN TEORI

### A. Cloud Computing

Menurut Mell [8], *Cloud computing* adalah sebuah model yang memungkikan sebuah kumpulan *computing resources* (jaringan, server, media penyimpanan, aplikasi, dan layanan) yang dapat dikonfigurasi, bisa diakses di manapun dengan mudah dan sesuai dengan keinginan. *Computing resources* tersebut dapat dengan cepat disediakan dan dilepaskan dengan usaha yang sedikit atau tidak sama sekali.

### B. Kontainer

Ranjan Dhar [9] menjelaskan beberapa implementasi awal dari teknologi kontainer. Pertama-tama, Ranjan menjelaskan soal Solaris Container, Solaris Container adalah sebuah implementasi dari teknologi virtualisasi di level sistem operasi untuk X86 dan sistem SPARC (*Scalable Processor Architecture*). Setelah itu, Ranjan menjelaskan tentang BSD Jails, BSD Jails dibangun diatas konsep chroot. Chroot sendiri adalah sebuah operasi untuk mengubah *root directory* dari sebuah proses, menyebabkan proses tersebut menjadi sebuah proses yang terisolasi dan aman.




Gambar 1. Aristektur Virtualisasi

Pada tahun 2008, Linux Container (LXC) dirilis walaupun untuk *mass deployment*-nya dilakukan pada tahun 2014 ketika kernel linux versi 3.8 dirilis. LXC adalah sebuah metode virtualisasi di level sistem operasi yang memungkinkan sebuah *host* menjalankan beberapa sistem linux yang terisolasi. Docker memakai LXC sebagai basisnya sebelum berganti ke libcontainer.

Claus Pahl [1] menjelaskan 3 ide dasar kontainerisasi. Ketiga ide dasar tersebut menjelaskan bahwa kontainer merupakan sebuah *runtime* yang ringan dan portabel, mempunyai kemampuan untuk mengembangkan, melakukan*testing*, dan melakukan *deploy* terhadap aplikasi ke banyak server, dan kemampuan untuk menghubungkan satu kontainer dengan kontainer lainnya. Gambar 2.1menggambarkan perbedaan virtualisasi antara teknologi VMdengan kontainer.

Menurut Miguel [10], kontainer (*container-based virtualization*) adalah sebuah arsitektur virtualisasi yang menjadi alternatif yang lebih ringan daripada virtualisasi yang arsitekturnya berbasis *hypervisor*. Karakteristiknya adalah dapat mempunyai banyak *user-spaces* terisolasi yang berjalan diatas sistem operasi. Kontainer juga menyediakan abstraksi langsung kepada proses *guest*.

### C. HTTP (HyperText Transfer Protocol)

Menurut standar RFC2616 [11], HTTP merupakan protkol di level aplikasi untuk sistem informasi *hypermedia* yang terdistribusi dan kolaboratif, HTTP bersifat generikdan *stateless* yang bisa digunakan untuk banyak hal selain untuk *hypertext*. Protokol ini sudah mulai digunakan semenjak tahun 1990 oleh *World-Wide Web global information initiative*.

### D. Load Balancer

Menurut Islam [12], *load balancer* adalah sebuah komponen yang penting dan berpengaruh besar di dalam sebuah infrastruktur jaringan dimana *resource* terdistribusi ke banyak sistem yang berbeda dan harus dibagikan kepada klien. Islam juga menjelaskan *load balancing*. *Load balancing* adalah sebuah cara untuk membagikan beban kerja ke banyak *resource*, dimana *load balancing* ini mempunyai tujuan akhir mengoptimalkan alokasi *resource*, memperkecil *response time*, dan mengurangi resiko *overload*.

### E. Load Testing

Menurut Daniel [13], performa sebuah infrastruktur IT dapat diukur menggunakan metode *load testing. Load testing* adalah sebuah cara untuk mengukur QoS (*Quality of Servi*ce) dari sebuah website. *Load testing* mempunyai *load generator* yang nantinya bertugas untuk menirukan perilaku sebuah *browser. Load generator* akan secara terus menerus mengirimkan *request* ke server yang dituju, menunggu beberapa saat setelah server mengirimkan balasan, lalu mengirimkan *request* baru.

Parameter yang dapat digunakan untuk mengukur performa sebuah infrastruktur IT adalah *availability* dan *response time*. *Availability* mengukur persentase pengakses yang berhasil mengakses infrastruktur IT



JUTEI Edisi Volume.5 No.1 April 2021
ISSN 2579-3675, e-ISSN 2579-5538
DOI 10.21460/jutei.2021.51.229

tersebut. Sedangkan untuk *response time* mengukur seberapa lama waktu yang dibutuhkan pengakses untuk mengakses infrastruktur IT.

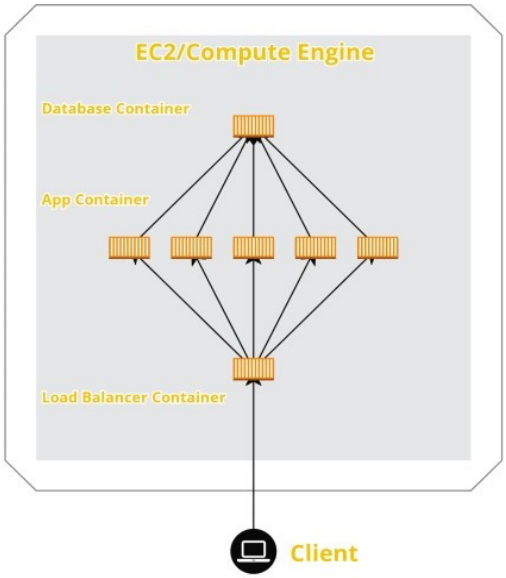
## III. METODOLOGI PENELITIAN

Penelitian dilakukan dengan cara membangun sistem di AWS dan GCP yang dapat menerima *request HTTP*. Sistem tersebut dibangun di atas *virtual machine* (VM) yang diatasnya di-*install* docker.

Penulis akan menggunakan laptop untuk melakukan simulasi trafik. Trafik akan dibuat menggunakan perangkat lunak jmeter. Parameter jmeter disesuaikan setiap kali penulis melakukan pengambilan data. Parameter jmeter yang akan disesuaikan setiap kali penulis melakukan pengambilan data adalah *number of threads*.


Gambar 2. Arsitektur Umum

Trafik akan dikirim ke *cloud provider*. Di *cloud provider*-nya sendiri, penulis akan memakai layanan *database* yang disediakan oleh *cloud provider* dan membuat sebuah VM di *cloud provider.* Untuk *database*-nya, penulis akan menggunakan layanan yang bernama RDS *relational database service* untuk di AWS dan Cloud SQL untuk di GCP. Untuk bagian VM-nya, RAM (*random access memory*) dan CPU (*central processing unit*) dari VM akan disesuaikan setiap kali penulis akan mengambil data. *Cloud provider* juga akan diganti dan disesuaikan setiap kali penulis melakukan pengambilan data.Di dalam VM tersebut akan di-*install* perangkat lunak docker. Diatas docker tersebut akan dibuat 5 buah kontainer yang berisi aplikasi, sebuah kontainer yang akan menjadi *load balancer,* dan sebuah kontainer MySQL yang akan menjadi *database* untuk aplikasi. *Load balancer* ini akan memakai NGINX dan HAPROXY.




Gambar 3. Arsitektur Kontainer dalam VM

Kontainer *load balancer* akan di *expose* ke luar docker menggunakan fitur *expose* yang disediakan oleh docker, sehingga kontainer *load balancer* dapat diakses dari luar. Inijuga berarti kontainer *load balancer* dapat menerima *request* dari luar. *Request* ini nantinya akan dibagikan oleh kontainer *load balancer* ke kontainer aplikasi yang berada didalam docker.

TABEL 1. TABEL PENGUJIAN

| Cloud Provider | Load Balancer | Instance Type | Number of Threads |
|---|---|---|---|
| AWS | NGINX | 1 vCPU, 2 GB | 100 |
| | | | 250 |
| | | | 500 |
| | | 2 vCPU, 4 GB | 100 |
| | | | 250 |
| | | | 500 |
| | | 4 vCPU, 16 GB | 100 |
| | | | 250 |
| | | | 500 |
| | HAPROXY | 1 vCPU, 2 GB | 100 |
| | | | 250 |
| | | | 500 |
| | | 2 vCPU, 4 GB | 100 |
| | | | 250 |
| | | | 500 |
| | | 4 vCPU, 16 GB | 100 |
| | | | 250 |
| | | | 500 |

| | | | |
|---|---|---|---|
| GCP | NGINX | 1 vCPU, 2 GB | 100 |
| | | | 250 |
| | | | 500 |
| | | 2 vCPU, 4 GB | 100 |
| | | | 250 |
| | | | 500 |
| | | 4 vCPU, 16 GB | 100 |
| | | | 250 |
| | | | 500 |
| | HAPROXY | 1 vCPU, 2 GB | 100 |
| | | | 250 |
| | | | 500 |
| | | 2 vCPU, 4 GB | 100 |
| | | | 250 |
| | | | 500 |
| | | 4 vCPU, 16 GB | 100 |
| | | | 250 |
| | | | 500 |

Tabel (1) menunjukkan kelas pengujian yang akan diuji oleh penulis.

## IV. HASIL DAN ANALISIS

### A. *Hasil Pengujian dari Sisi Response Time*

Berikut ini adalah hasil rata-rata *response time* yang didapatkan dari pengujian yang telah dilakukan oleh penulis.Penelitian ini dilakukan di jaringan yang stabil.

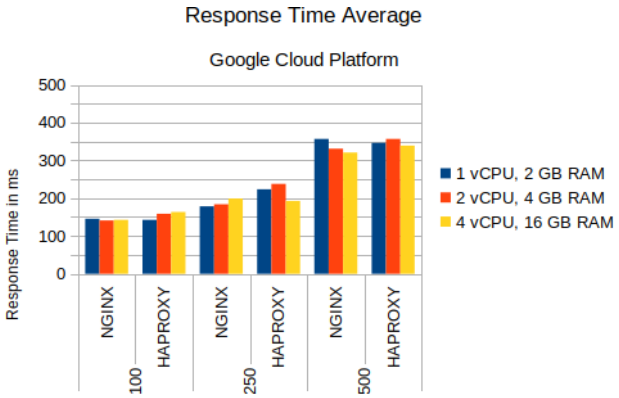
TABEL 2. RATA-RATA RESPONSE TIME DI GCP (MS)

| **Spesifikasi** | | **NGINX** | | | **HAPROXY** | | |
|---|---|---|---|---|---|---|---|
| **vCPU** | **RAM** | **100** | **250** | **500** | **100** | **250** | **500** |
| 1 | 2 | 145.55 | 178.45 | 356.77 | 142.64 | 223.61 | 346.54 |
| 2 | 4 | 141.17 | 183.96 | 330.91 | 158.77 | 237.98 | 356.75 |
| 4 | 16 | 142.75 | 199.02 | 320.82 | 163.54 | 192.75 | 339.16 |

TABEL 3. RATA-RATA RESPONSE TIME DI AWS (MS)

| **Spesifikasi** | | **NGINX** | | | **HAPROXY** | | |
|---|---|---|---|---|---|---|---|
| **vCPU** | **RAM** | **100** | **250** | **500** | **100** | **250** | **500** |
| 1 | 2 | 136.45 | 174.49 | 258.31 | 147.45 | 191.27 | 310.84 |
| 2 | 4 | 143.13 | 178.26 | 299.54 | 134.87 | 167.53 | 278.35 |
| 4 | 16 | 146.79 | 193.39 | 305.34 | 144.53 | 185.4 | 326.6 |

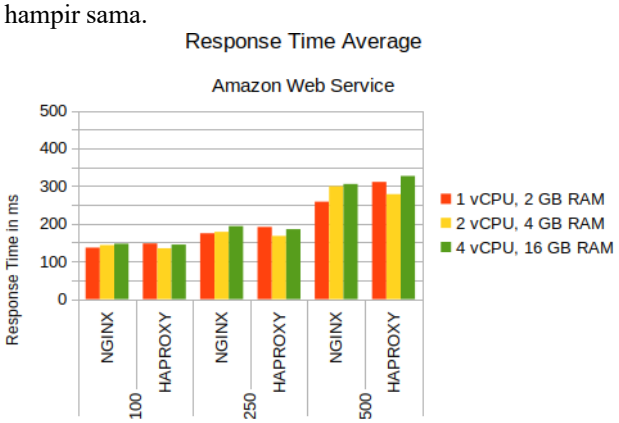
Berikut adalah grafik rata-rata *response time* yang didapatkan oleh penulis.




Gambar 4. Grafik rata-rata response time di GCP

Gambar (4) menunjukkan grafik rata-rata *response time* dari hasil pengujian yang penulis lakukan di *cloud platform* Google Cloud Platform. *Load balancer* NGINX dan HAPROXY di tiap kelas mempunyai *response time* yang hampir sama.

Gambar (5) menunjukkan grafik rata-rata response time dari hasil pengujian yang dilakukan oleh penulis di *cloud platform* AWS. Grafik ini menunjukkan bahwa setiap kelas ketika menggunakan spesifikasi *virtual machine* yang berbeda-beda mempunyai *response time* yang hampir sama.

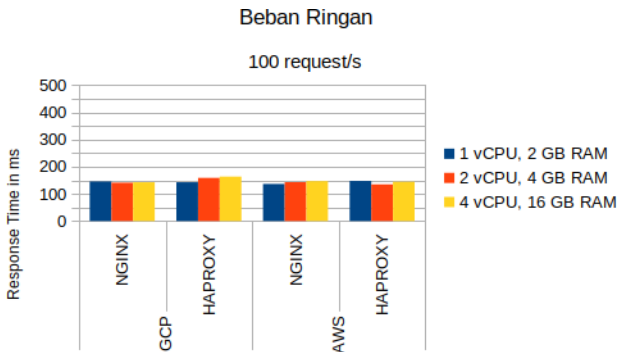



Gambar 5. Grafik rata-rata response time di AWS

Grafik di (4) dan (5) menunjukkan bahwa *response time* di masing-masing beban dan *load balancer* tidak dipengaruhi oleh spesifikasi *virtual machine* ketika beban trafik yang diberikan 100, 250, dan 500 *request* per detik. Kedua grafik tersebut juga menunjukkan bahwa semakin tinggi beban yang diberikan, maka selisih antara satu percobaan dengan percobaan yang lain yang mempunya beban yang sama semakin besar. Hal ini dapat disebabkan oleh faktor kestabilan jaringan ketika pengujian dilakukan, semakin besar beban yang dikirim maka semakin besar pula faktor kestabilan jaringan mempengaruhi hasil pengujian.



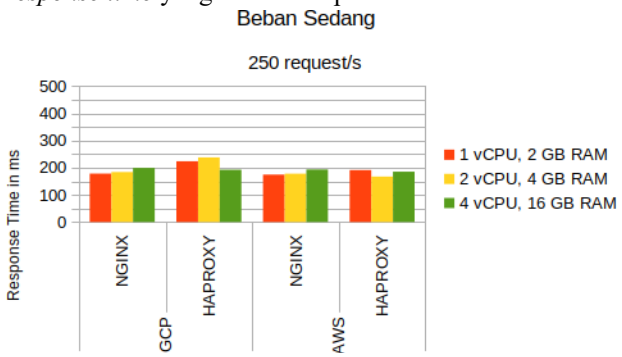
JUTEI Edisi Volume.5 No.1 April 2021
ISSN 2579-3675, e-ISSN 2579-5538
DOI 10.21460/jutei.2021.51.229




Gambar 6. Grafik rata-rata response time dengan beban ringan

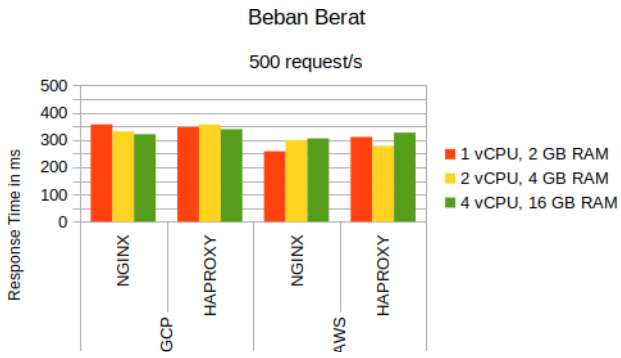
Gambar (6) menunjukkan grafik rata-rata *response time* dengan beban ringan untuk setiap jenis *load balancer* dan *cloud provider*. Ketika pengujian dilakukan dengan beban ringan, maka setiap jenis *load balancer* dari masing-masing *cloud provider* mempunyai *response time* yang hampir sama.

Gambar (7) menunjukkan grafik rata-rata *response time* dengan beban sedang. Grafik menunjukkan bahwa *load balancer* HAPROXY di AWS mempunyai *response time* yang lebih kecil dibanding *load balancer* HAPROXY diGCP. Hal ini juga berlaku di seluruh jenis spesifikasi *virtual machine.* Sedangkan untuk *load balancer* NGINX, kedua *cloud provider* menunjukkan *response time* yang relatif mirip.




Gambar 7. Grafik rata-rata response time dengan beban sedang

Gambar (7) menunjukkan grafik rata-rata *response time* dengan beban berat di setiap *load balancer* dan *cloud provider*. Grafik menunjukkan bahwa untuk setiap spesifikasi *virtual machine* dan setiap jenis *load balancer* di AWS mempunyai *response time* yang lebih kecil daripada *response time* yang dimiliki oleh GCP.




Gambar 8. Grafik rata-rata response time dengan beban berat

Gambar (5), (6), dan (7) menunjukkan bahwa jenis *load balancer* dan *cloud provider* mempunyai pengaruh terhadap *response time* ketika pengujian dilakukan dengan beban sedang dan berat. AWS mempunyai *response time* yanglebih kecil dibandingkan dengan GCP ketika beban yang diberikan berat. Untuk perihal HAPROXY di AWS yang mempunyai *response time* yang lebih kecil dibanding HAPROXY di GCP ketika pengujian dilakukan dengan

### *B. Hasil Pengujian dari Sisi Error Rate*

Berikut ini adalah hasil *error rate* yang didapatkan dari pengujian yang telah dilakukan oleh penulis. Penelitian ini dilakukan di jaringan yang stabil.

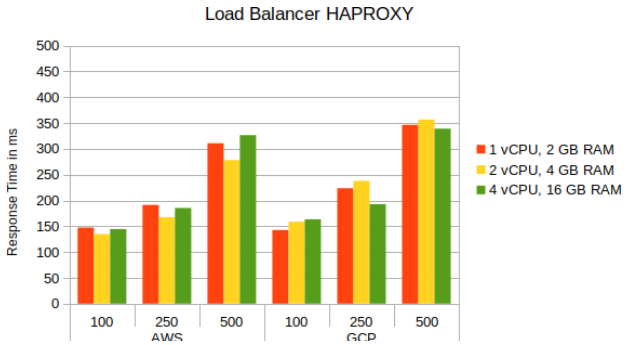
TABEL 4. JUMLAH ERROR DI GCP

| Spesifikasi | | NGINX | | | HAPROXY | | |
|---|---|---|---|---|---|---|---|
| vCPU | RAM | 100 | 250 | 500 | 100 | 250 | 500 |
| 1 | 2 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2 | 4 | 0 | 0 | 0 | 0 | 0 | 0 |
| 4 | 16 | 0 | 0 | 0 | 0 | 0 | 0 |

TABEL 5. JUMLAH ERROR DI AWS

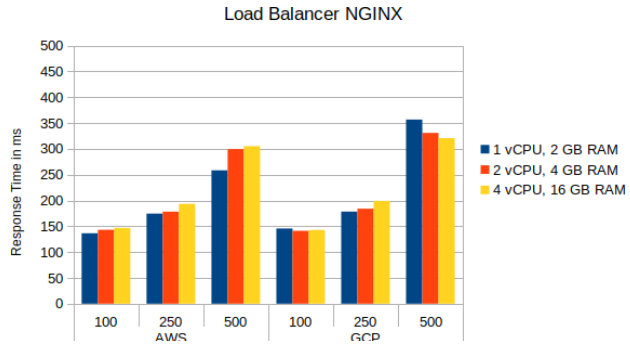
| Spesifikasi | | NGINX | | | HAPROXY | | |
|---|---|---|---|---|---|---|---|
| vCPU | RAM | 100 | 250 | 500 | 100 | 250 | 500 |
| 1 | 2 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2 | 4 | 0 | 0 | 0 | 0 | 0 | 0 |
| 4 | 16 | 0 | 0 | 0 | 0 | 0 | 0 |

menggunakan beban sedang diperjelas dengan grafik berikut.




Gambar 9. Grafik rata-rata response time di HAPROXY

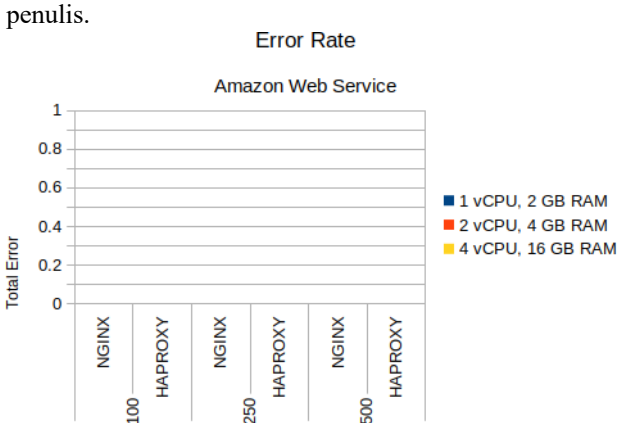
Gambar (8) menunjukkan grafik rata-rata *response time* dengan jenis *load balancer* HAPROXY. Grafik menunjukkan bahwa ketika beban yang diberikan sedangdan berat, AWS mempunyai *response time* yang lebih kecil dibanding GCP untuk setiap spesifikasi *virtual machine*.



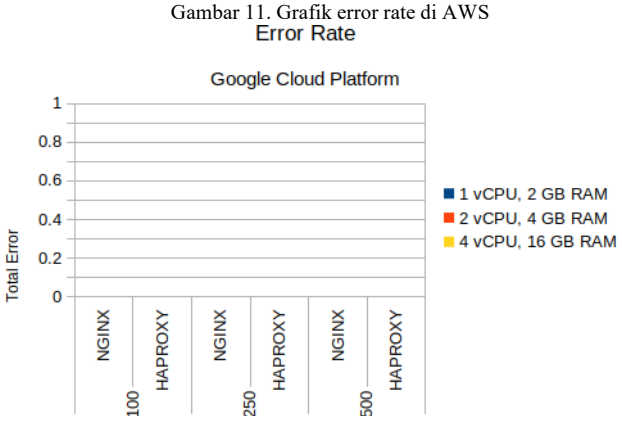

Gambar 10. Grafik rata-rata response time di NGINX

Gambar (9) menunjukkan grafik rata-rata *response time* dengan jenis *load balancer* NGINX. Grafik menunjukkan bahwa AWS mempunyai *response time* yang lebih kecil dibandingkan dengan GCP ketika pengujian dilakukandengan menggunakan beban berat. Hal ini berlaku untuk setiap spesifikasi *virtual machine*.

Berikut adalah grafik *error rate* yang didapatkan oleh penulis.




Gambar 11. Grafik error rate di AWS




Gambar 12. Grafik error rate di GCP

Kedua grafik tersebut menunjukkan bahwa GCP dan AWS mampu melayani semua *request* tanpa *error*. Terlepas dari faktor jenis *load balancer* yang digunakan, beban yang diberikan, dan spesifikasi *virtual machine* yang digunakan.

### C. *Perbandingan Jaringan*

Penulis melakukan ulang pengujian tetapi menggunakan jaringan yang berbeda. Pengujian ini dilakukan untuk membuktikan pengaruh jaringan terhadap hasil pengujian. Perlu diingat, jaringan yang dipakai dalam penelitian ini mempunyai *throughput* yang tidak stabil.

TABEL 6. RATA-RATA RESPONSE TIME DI GCP (MS) MENGGUNAKAN JARINGAN YANG TIDAK STABIL

| Spesifikasi | | NGINX | | | HAPROXY | | |
|---|---|---|---|---|---|---|---|
| vCPU | RAM | 100 | 250 | 500 | 100 | 250 | 500 |
| 1 | 2 | 193.18 | 237.68 | 473.13 | 167.83 | 216.58 | 277.01 |
| 2 | 4 | 158.76 | 204.26 | 416.66 | 152.37 | 200.26 | 337.77 |
| 4 | 16 | 170.74 | 210.72 | 400.57 | 165.07 | 196.86 | 419.47 |

TABEL 7. RATA-RATA RESPONSE TIME DI AWS (MS) MENGGUNAKAN JARINGAN YANG TIDAK STABIL

| Spesifikasi | | NGINX | | | HAPROXY | | |
|---|---|---|---|---|---|---|---|
| vCPU | RAM | 100 | 250 | 500 | 100 | 250 | 500 |
| 1 | 2 | 179.79 | 235.48 | 677.48 | 263.22 | 499.64 | 430.73 |
| 2 | 4 | 185.68 | 231.63 | 376.24 | 208 | 202.42 | 436.18 |
| 4 | 16 | 226.08 | 260.06 | 423.02 | 151.86 | 215.91 | 358.75 |

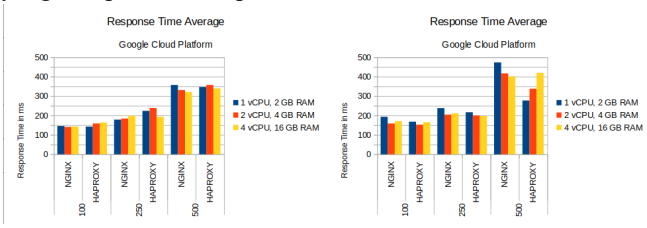
TABEL 8. JUMLAH ERROR DI GCP MENGGUNAKAN JARINGAN YANG TIDAKSTABIL

| Spesifikasi | | NGINX | | | HAPROXY | | |
|---|---|---|---|---|---|---|---|
| vCPU | RAM | 100 | 250 | 500 | 100 | 250 | 500 |
| 1 | 2 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2 | 4 | 0 | 0 | 0 | 0 | 0 | 0 |
| 4 | 16 | 0 | 0 | 0 | 0 | 0 | 0 |

TABEL 9. JUMLAH ERROR DI AWS MENGGUNAKAN JARINGAN YANG TIDAKSTABIL

| Spesifikasi | | NGINX | | | HAPROXY | | |
|---|---|---|---|---|---|---|---|
| vCPU | RAM | 100 | 250 | 500 | 100 | 250 | 500 |
| 1 | 2 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2 | 4 | 0 | 0 | 0 | 0 | 0 | 0 |
| 4 | 16 | 0 | 0 | 0 | 0 | 0 | 0 |

Berikut adalah grafik rata-rata *response time* dari data yang didapatkan oleh penulis.




Gambar 13. Perbandingan grafik rata-rata response time di GCP antara jaringan normal (kiri) dan jaringan yang tidak stabil (kanan)

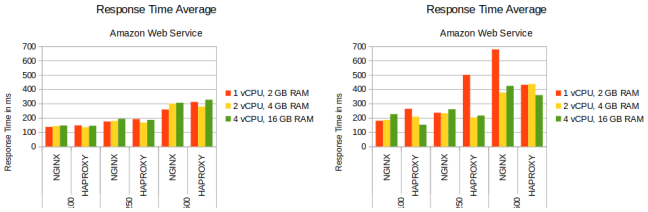
Gambar (13) menunjukkan perbandingan grafik rata-rata *response time* di GCP ketika menggunakan jaringan normal dan jaringan yang tidak stabil. Grafik hasil uji ketika menggunakan jaringan yang tidak stabil memperlihatkan selisih *response time* yang besar di antara spesifikasi *virtual machine* yang sama yang menggunakan beban dan jenis *load balancer* yang sama bila



JUTEI Edisi Volume.5 No.1 April 2021
ISSN 2579-3675, e-ISSN 2579-5538
DOI 10.21460/jutei.2021.51.229

dibandingkan dengan grafik

hasil uji yang menggunakan jaringan normal. Grafik di gambar (13) juga menunjukkan bahwa semakin besar beban




Gambar 14. Perbandingan grafik rata-rata response time di AWS antarajaringan normal (kiri) dan jaringan yang tidak stabil(kanan)

yang dipakai, maka akan semakin besar selisih *response time*.

Gambar 4.22 menunjukkan perbandingan grafik rata-rata *response time* di AWS yang pengujiannya menggunakanjaringan normal dan jaringan yang tidak stabil. Grafik ini memperlihatkan hal yang sama seperti gambar 4.21 dengan beberapa pengecualian. Pengujian di AWS mempunyai beberapa data yang *response time*-nya tidak mirip dengan kelas yang sama dengan data tersebut seperti pengujian yangdilakukan dengan beban sedang di HAPROXY. *virtual machine* dengan spesifikasi 1 vCPU dan 2 GB RAM mempunyai *response time* lebih dari 500ms, sedangkan *virtual machine* dengan spesifikasi 2 vCPU dan 4 GB RAM,dan 4 vCPU dan 16 GB RAM mempunyai rata-rata *response time* dibawah 250 ms.

## V. KESIMPULAN DAN SARAN

### *A. Kesimpulan*

Berdasarkan hasil yang didapatkan, maka dapat disimpulkan hal-hal berikut:

1. AWS dapat menangani beban sedang dan berat lebih baik daripada GCP.
2. Spesifikasi *virtual machine* tidak berpengaruh terhadap *response time* ketika beban yang diberikatdibawah 500 *request* per detik.
3. Semakin besar trafik yang diberikan, maka pengaruh kestabilan jaringan juga semakin besar.
4. Berdasarkan konfigurasi yang sudah penulis buat, *load balancer* NGINX dapat menangani beban sedang dan berat lebih baik daripada HAPROXY.
5. GCP dan AWS dapat menangani beban ringan, sedang, dan berat dengan baik jika dilihat dari sisi jumlah *error*.

### *B. Saran*

Penulis menyarankan beberapa hal untuk pengembanganpenelitian kedepan, beberapa saran tersebut adalah:

1. Menggunakan konfigurasi yang berbeda untuk masing-masing HAPROXY dan NGINX.
2. Membandingkan algoritma *load balancing* yangberbeda.
3. Menggunakan *database* diluar *virtual machine*.
4. Menggunakan aplikasi yang lebih berat.
5. Menggunakan beban trafik yang lebih bervariatifdan berat.
6. Menggunakan *cloud provider* lain seperti MicrosoftAzure dan Alibaba Cloud.
7. Membuat *virtual machine* di lokasi yang lebih dekat dengan pengujian.

## DAFTAR PUSTAKA

[1] CPahl, “Containerization and the PaaS Cloud,” IEEE Cloud Comput., vol. 2, no. 3, pp. 24–31, 2015.

[2] Diamanti, “2018 Container Adoption,” 2018.

[3] R. Anandhi and K. Chitra, “A Challenge in Improving the Consistency of Transactions in Cloud Databases - Scalability,” *Int. J. Comput. Appl.*, vol. 52, no. 2, pp. 12–14, 2012.

[4] K. Salchow, “Load balancing 101: Nuts and bolts,” *F5 Networks, Inc.*, pp. 1–6, 2007.

[5] S. Dadi, “Evaluasi Metode Load Balancing dan Fault Tolerance pada Chatting Server Jaringan Sosial,” 2017.

[6] M. W. I. Santosa, R. Primananda, and W. Yahya, “Implementasi Load Balancing Server Basis Data Pada Virtualisasi Berbasis Kontainer,” Malang, 2018.

[7] F. N. A. Majid, “Studi Komparasi Performa Load Balancer dengan Menggunakan Teknologi Docker Container,” Yogyakarta, 2019.

[8] P. Mell and T. Grance, “The NIST-National Institute of Standarsand Technology- Definition of Cloud Computing,” *NIST Spec. Publ. 800- 145*, p. 7, 2011.

[9] R. Dhar, “Comparative evaluation of Virtual Environments: Virtual Machines and Containers,” *J. Ment. Sci.*, 2016.

[10] M. G. Xavier, M. V Neves, and C. A. F. De Rose, “A Performance Comparison of Container-Based Virtualization Systems for MapReduce Clusters,” no. February, 2014.

[11] R. Waterman *et al.*, “Remote Network Monitoring MIB Extensions for Switched Networks,” 1999.

[12] S. Islam, “Network Load Balancing Methods : Experimental Comparisons and Improvement,” *arXiv Prepr. arXiv1710.06957.*, 2017.

[13] D. A. Menascé, “Load testing of Web sites,” *IEEE Internet Comput.*, vol. 6, no. 4, pp. 70–74, 2002