Available *online* at : http://bit.ly/InfoTekJar

InfoTekJar : Jurnal Nasional Informatika dan Teknologi Jaringan

ISSN (Print) 2540-7597 | ISSN (Online) 2540-7600

Analisis Performansi *High Availability Web Server* Pada *Cluster* GKE (*Google Kubernetes Engine*) Menggunakan Infrastruktur *Google Cloud Platform*

Jafaruddin Gusti Amri Ginting, Syariful Ikhwan, Muhammad Naufal Ammar

Institut Teknologi Telkom Purwokerto, Jl D.I.Panjaitan No.128 Purwokerto 53147, Indonesia

KATA KUNCI

Web server, High Availability, Cluster, docker, Kubernetes, Cloud Computing

KORESPONDENSI

Phone: +62 813 7484 8381

E-mail: syariful@ittelkom-pwt.ac.id

ABSTRAK

Pengguna internet di dunia menurut *Internet World* Stat pada tahun 2020 mencapai 58.7% dari pengguna internet diseluruh dunia, Persentase tersebut tersebut meningkat dari tahun ke tahun berikutnya. Pertumbuhan pengguna internet diseluruh dunia yaitu 1.167% dari tahun 2000 sampai tahun 2020. Banyaknya pengguna internet membuat penyedia layanan aplikasi web harus memikirkan ketersediaan *resource* untuk web server yang dikelola. Berdasarkan masalah tersebut, diperlukan suatu sistem yang dapat membantu kinerja dan *availability* server. Server *clustering* memungkinkan sejumlah komputer untuk bekerja sama melakukan proses komputasi. Teknologi server clustering memiliki kelebihan dapat menghasilkan suatu sistem dengan tingkat keandalan dan ketersediaan yang tinggi. Kubernetes merupakan salah satu platform cluster untuk container atau *container orchestrator*. Kubernetes diharapkan dapat menjadi solusi agar proses komputasi lebih efisien dan terciptanya sistem dengan tingkat ketersediaan yang tinggi. Sistem disimulasikan menggunakan *platform public cloud* yaitu *google cloud platform* dengan service *google kubernetes engine*. Sistem disimulasikan menggunakan *google kubernetes engine* pada *google cloud platform*. Nilai rata-rata *Availability* diperoleh 99.96% dari total 27526 menit *uptime*. Dengan beban komputasi *node* pada *google kubernetes engine* rata-rata CPU *usage* setiap skenario pengujian didapatkan cukup stabil dengan CPU tertinggi sebesar 27.178%. Dari hasil pengujian *throughput*, didapatkan nilai rata-rata *throughput* pada setiap koneksi sebesar 7.126 Mbit/s dengan nilai tertinggi pada 1000 koneksi yaitu 8.764 Mbit/s. *Delay* menunjukkan kategori "Sangat Baik" berdasarkan standarisasi TIPHON yaitu kurang dari 150 ms. Persentase *packet loss* paling tinggi didapat pada 5000 koneksi sebesar 16.27%.

PENDAHULUAN

Pada era digital penggunaan aplikasi berbasis web cenderung meningkat seiring banyaknya pengguna internet diseluruh dunia. Dari data statistik pengguna internet secara individual menurut ITU pada paper 2019 *Measuring digital development Facts and figures* 2019, pada tahun tersebut rata-rata pengguna internet individualnya mencapai 53.6% diseluruh dunia. Benua eropa menjadi benua dengan persentase paling besar 82.5% dan Asia menempati posisi ke-5 dengan persentase 48.4% [1]. Mulai banyaknya permintaan request jumlah pengguna internet menjadikan penyedia layanan atau perusahaan aplikasi berbasis web harus meningkatkan dari segi *resource* mereka terutama server, semakin meningkat jumlah permintaan request maka akan mengakibatkan server *down* atau *down time* karena server *overload*. Teknologi *Cloud computing* dibutuhkan sebagai teknologi komputasi terdistribusi yang mampu mengabstraksikan kemampuan hardware atau perangkat keras dalam menjalankan sebuah proses komputasi atau sistem operasi, efisiensi *resource* penggunaan perangkat dari segi *hardware* dan proses untuk *high availability*, serta sistem yang lebih handal untuk menangani sistem kegagalan.[2] Perkembangan teknologi virtualisasi juga sebagai penyebab berkembangnya teknologi *Cloud computing* dan kebutuhan proses abstraksi proses komputasi yang dibutuhkan oleh perusahaan. Dalam implementasi teknologi virtualisasi tradisional dirasa terlalu memakan banyak *resource* perangkat keras untuk membangun atau menjalankan sebuah proses komputasi. Maka solusinya untuk menerapkan virtualisasi komputasi berbasiskan teknologi *container* yang lebih minim dalam penggunaan *resource* dan ringan. Perusahaan penyedia layanan berbasis web atau penyedia layanan aplikasi terbesar di dunia seperti *Google*, *Microsoft* dan *Amazon* menggunakan teknologi virtualisasi *container* untuk menjalankan sistem terdistribusinya. Meskipun 1 teknologi *container* sudah ada dan dikenal sejak lama tetapi banyak perusahaan yang baru ingin menerapkan teknologi tersebut dikarenakan memiliki rasa kekhawatiran atas kinerja. Akan tetapi performa dari teknologi

https://doi.org/10.30743/infotekjar.v5i2.3577

Attribution-NonCommercial 4.0 International. Some rights reserved

container sudah hampir tidak masalah lagi karena teknologi *container* yang ada saat ini semakin berkembang pesat karena banyaknya *project* yang berbasis *open source* dengan memanfaatkan teknologi *container*. Teknologi ini memiliki banyak keuntungan seperti untuk kemudahan membuat *services* aplikasi yang banyak dan cepat [3]. Aplikasi perangkat lunak Web server merupakan aplikasi untuk menjalankan dan mengeksekusi teknologi *web world wide web* (www). Web server bekerja ketika ada permintaan dari *client* yang menggunakan browser semisal, *Internet Explorer, Mozilla,* dan aplikasi *browser* lainnya. Jika ada permintaan dari *browser client* maka web server akan memproses permintaan *client* kemudian memberikan hasil berupa data yang diinginkan kembali ke *client browser* [4]. Metode yang digunakan server untuk berkomunikasi dengan protokol TCP *three-way handshake*. Teknik *handshaking* TCP disebut juga SYN-SYN-ACK dan SYN (*Synchronous*), SYN-ACK (*Synchronous-Acknowledge*), ACK (*Acknowledge*) karena ada tiga pesan yang akan dikirimkan oleh TCP untuk berkomunikasi antara *client* dan Server. *Client* mengirimkan pesan SYN ke web server lalu web server membalasnya dengan dengan pesan SYN-ACK ke client, kemudian client membalas lagi dengan pesan ACK. Tugas dari web server adalah menanggapi setiap permintaan oleh *client* atau web *browser*. Semakin banyaknya permintaan yang ditujukan ke web server, maka *resource* yang digunakan semakin banyak [5].

Variabel yang akan diamati pada penelitian ini antara lain: pengujian *availability ,throughput, delay, packet loss* dan CPU *Usage* Nilai-nilai tersebut didapatkan melalui perhitungan terhadap data hasil *capture packet* menggunakan *software* Wireshark setelah dilakukan pengujian terhadap aplikasi *web server* ditambah Wordpress menggunakan *software* HTTPerf. Adapun tujuan penelitian ini yaitu menganalisis parameter QoS yang dihasilkan dari pengujian akses dari pengguna serta melakukan kinerja ketersedian aplikasi web server pada *cluster Google kubernetes engine.*

Web server dapat merujuk ke perangkat keras atau perangkat lunak, atau keduanya dapat bekerja sama. Di sisi perangkat keras, web server merupakan komputer yang menyimpan file komponen situs web misalnya, Dokumen HTML, gambar, *stylesheet* CSS dan file Javascript. Situs web yang berisi file komponen tersebut terhubung ke internet dan mendukung pertukaran data antar client dan server. Di sisi perangkat lunak, web server mencakup beberapa bagian yang mengontrol cara pengguna atau *client* dapat mengakses file web yang ada di server, Dengan menggunakan protokol HTTP/s. Server HTTP adalah perangkat lunak yang memahami URL (alamat web) dan HTTP (protokol yang digunakan *browser client* untuk melihat halam web). Dan di sisi perangkat lunak kita bisa mengakses web server dengan nama domain seperti google.com [6].

Kubernetes merupakan *platform cluster open source* yang digunakan untuk otomatisasi dari segi deployment, scaling dan manajemen container. Kubernetes merupakan *platform* yang *portable* karena dapat diaplikasikan pada *private* maupun *public cloud* [7]. Selain itu kubernetes hanya menggunakan *resources hardware* yang dibutuhkan sehingga penggunaan *resources* lebih optimal.

Cloud Computing merupakan teknologi yang menyediakan sumber daya seperti, Storage, Operating System, Virtual Machine yang biasa kita kenal VPS (Virtual Private Server). Dengan menggunakan layanan Cloud Computing dimana pengguna hanya membayar services yang sedang berjalan konsep ini dinamakan pay-as-you-go atau basis pembayaran tergantung pemakaian. Perusahaan yang menyediakan layanan tersebut disebut Cloud Provider. Beberapa penyedia contoh adalah Microsoft, Amazon, Google, dan Alibaba. Cloud Provider bertanggung jawab atas perangkat keras yang diperlukan untuk menjalankan aplikasi pengguna [8]. model deployment yang akan digunakan Cloud computing yaitu:

A. Public Cloud Google Cloud Platform

Model ini dimana seluruh sumberdaya perangkat keras yang menyediakan adalah Cloud provider.

Pada tahap ini akan dilakukan pengujian High availability dan performa dari simulasi clustering menggunakan kubernetes. Pengujian dilakukan dari sisi cluster dan client. Availability diukur berdasarkan istilah "nine", dimana semakin banyak 'nine' maka semakin tinggi sebuah ketersediaan atau availability pada suatu sistem [9]. Pengujian availability pada penelitian mengamati apakah layanan server tetap berjalan saat terjadi kegagalan pada salah satu nodes didalam cluster google kubernetes engine. pengujian akan menggunakan tools httperf. Pengujian availability pada penelitian ini mengamati apakah layanan server tetap berjalan saat terjadi kegagalan pada salah satu nodes didalam cluster kubernetes Perhitungan availability cluster berdasarkan rumus yang digunakan oleh Berdasarkan rumus menghitung availability ditunjukkan pada persamaan 1

$$Availability = \frac{MTBF}{MTBF + MTTR} \text{X } 100 \quad (1)$$

QoS merupakan suatu metode pengukuran yang digunakan untuk mengetahui kemampuan suatu jaringan agar dapat memberikan layanan *network* yang lebih baik dan terencana [10].

B. Throughput

Throughput merupakan nilai rata-rata pengiriman yang sukses dalam interval waktu tertentu yang diukur dalam satuan *bit per second* (bps atau bit/s). Rumus menghitung *throughput* ditunjukkan pada persamaan 2.

$$Throughput = \frac{Jumlah\ packet\ data\ yang\ dikirim\ (bit)}{Waktu\ pengiriman\ paket\ (second)} \quad (2)$$

C. Delay

Delay adalah waktu yang diperlukan sebuah paket untuk sampai ke penerima dimulai dari sisi pengirim. Rumus menghitung *delay* ditunjukkan pada persamaan 3 [10].

$$Delay\ rata - rata = \frac{Total\ delay}{Total\ paket\ yang\ diterima} \quad (3)$$

Pengelompokan standarisasi *delay* berdasarkan TIPHON ditunjukkan pada Tabel 1.

Tabel 1. Standarisasi *Delay*.

No	Kategori	Besar (ms)
1	Sangat Baik	< 150
2	Baik	150 – 249
3	Cukup Baik	250 – 349
4	Buruk	350 - 449
5	Tidak Direkomendasikan	> 449

D. Packet Loss

Packet loss merupakan parameter yang menunjukkan jumlah paket yang hilang. Rumus menghitung *packet loss* ditunjukkan pada persamaan 4 [10].

$$Pack.Loss = \frac{(Paket\ dikirim - paket\ diterima) x 100\%}{Paket\ dikirim} \quad (4)$$

Pengelompokan standarisasi *packet loss* berdasarkan TIPHON ditunjukkan pada Tabel 2.

Tabel 2. Standarisasi *Packet Loss*.

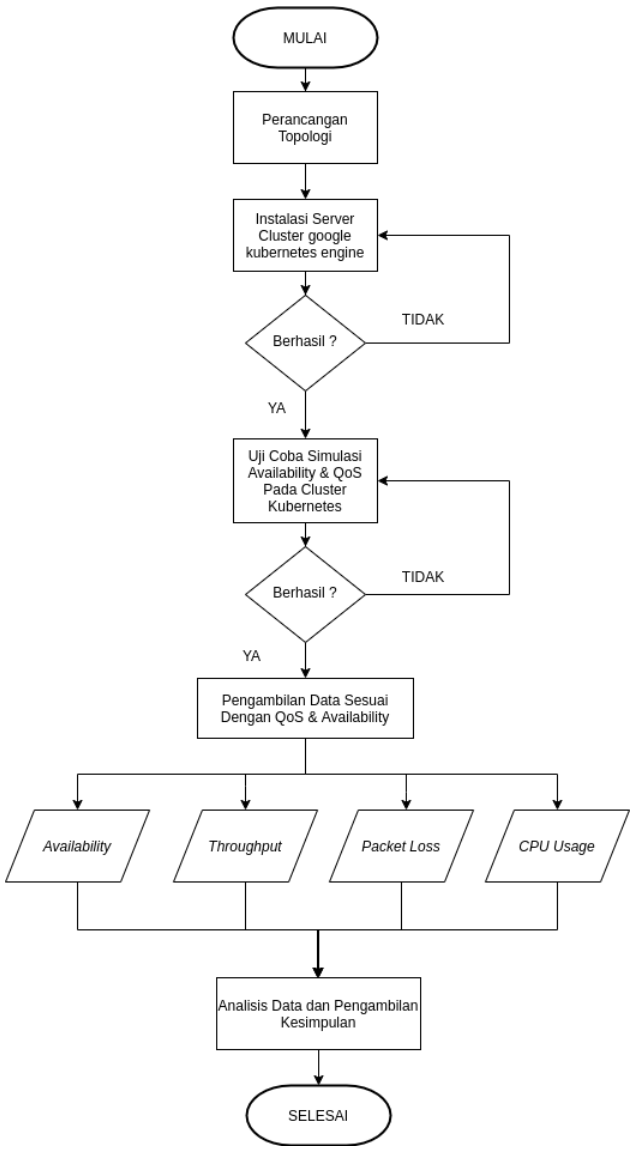
No	Kategori	Besar (%)
1	Sangat Baik	0 – 2
2	Baik	3 – 14
3	Cukup Baik	15 – 25
4	Buruk	> 25

METODOLOGI

Pada Penelitian ini digunakan metode penelitian eksperimental untuk mencari efek atau pengaruh perlakuan yang diberikan terhadap hal lainnya dalam kondisi yang terkendalikan. Efek atau pengaruh perlakuan yang dimaksud penulis adalah pengaruh penggunaan Kubernetes Orchestrator untuk menjalankan aplikasi web server dengan High availability. Prinsip yang digunakan pre-experimental dengan one shot case study, Control Experiment akan berisi pengukuran performa dari web server menggunakan google kubernetes engine. Percobaan kelompok akan menjalankan aplikasi web menggunakan cluster kubernetes dengan aplikasi cloud computing google kubernetes engine. Control Experiment akan digunakan sebagai acuan dalam penelitian ini untuk mendapatkan hasil performansi yang akan melakukan implementasi high availability dan implementasi cluster pada container dengan google kubernetes engine. Pada perancangan jaringan ini akan dilakukan teknik clustering menggunakan container orchestrator kubernetes dan docker sebagai container runtime yang akan menjalankan aplikasi atau service.

A. Konfigurasi Perangkat

Tahapan-tahapan penelitian digambarkan melalui diagram alur berikut.



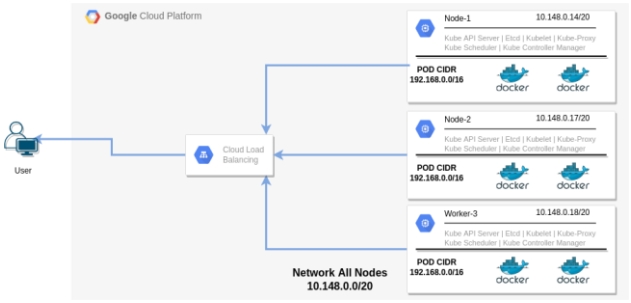
Gambar 1. Diagram Alur Penelitian

Gambar 1 merupakan tahapan perancangan dan analisis yang akan dilakukan selama penelitian. Pertama memulai dengan merancang topologi dan perancangan infrastructure server yang menggunakan Cluster kubernetes, jika berhasil selanjutnya melakukan simulasi high availability pada cluster kubernetes, jika simulasi berhasil akan dilanjutkan dengan Pengambilan data sesuai dengan QoS seperti Throughput, Delay dan Packet Loss pada setiap Cluster kubernetes diuji dari client menggunakan jaringan provider telkomsel berkecepatan 15-20 Mbps menuju aplikasi web server, Selanjutnya menguji ketersediaan atau High availability. Tahapan terakhir analisis data memasukan hasil data dan memberikan 18 kesimpulan dari hasil performa uji coba web server yang diimplementasikan di cluster kubernetes

Perancangan topologi menggunakan satu cluster kubernetes dengan tiga buah node yang berperan sebagai master dan worker. Node master akan menjalankan komponen api-server, scheduler, controller dan dashboard. Node worker akan menjalankan container runtime docker engine, kubelet dan kube-proxy. Node worker menjalankan container yang berisi service web server dan

 https://doi.org/10.30743/infotekjar.v5i2.3577

database server, Client mengakses dashboard yang telah diinstal dalam node master.



Gambar 2. Topologi Jaringan

Pada gambar 2 merupakan topologi jaringan dalam kubernetes. Node master merupakan pusat kendali dari node worker. Tiap user dapat memberikan perintah yang kemudian dieksekusi pada node master. Node master akan membagi tugas dari perintah yang dilakukan oleh client ke node worker. Kemudian node worker akan menjalankan perintah oleh user. Berikut merupakan kebutuhan perangkat keras dan perangkat lunak untuk menjalankan simulasi. Berikut kebutuhan perangkat yang digunakan pada penelitian ini pada Tabel 4

Tabel 4. Spesifikasi dan IP *Address* Perangkat

Perangkat	Spesifikasi	Keterangan
Client	OS	Elementary OS 5.1.6 Hera
	CPU	4
	RAM	8 GB
	Tool dan Aplikasi	1. kubectlv1.17.15 2. httperf 0.9.0 3. Wireshark 2.6.1 4. Speed Internet 20 Mbps
Node Master & Worker	OS	Ubuntu Server 18.04
	vCPU	2
	RAM	8 GB
	Tool dan Aplikasi	1. Kubernetes components 2. Wordpress 5.6 3. MySQL 5.7 4. GKE (1.17.15-gke.800)

B. Instalasi Dan Konfigurasi Google Kubernetes Engine

Sebelum melakukan pengujian, peneliti terlebih dahulu melakukan setup region *cluster* server yang akan digunakan Dalam Instalasi kubernetes pada google cloud platform, diperlukan beberapa repository utama-nya yaitu milik google karena kubernetes dikembangkan oleh google. Untuk memulai suatu cluster kubernetes kita harus inisialisasi node yang berperan sebagai master dan worker, dan menginstall gcloud. Gcloud digunakan sebagai layanan untuk menjalankan perintah pada layanan google cloud platform. Dan overlay network menggunakan flannel agar setiap pod container pada kubernetes cluster dapat terkoneksi melalui 3 mode networking yaitu ClusterIP yang dapat berkomunikasi Internal saja antar node cluster, NodePort dapat mengakses aplikasi pada pods cluster dengan port lalu Loadbalancer dimana aplikasi di dalam pods cluster kubernetes dapat diakses secara public. Untuk membuat area cluster kubernetes kita harus membuat area project di dalam public cloud google cloud platform dengan menginisialisasi nama project, region atau wilayah server yang akan digunakan, konfigurasi tersebut ditunjukan pada gambar 3.

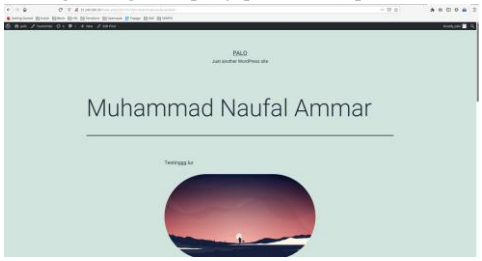
```
$ gcloud container clusters create "wp-cluster" \
--cluster-version "1.17.15-gke.800" \
--region "asia-southeast1" \
--image-type "UBUNTU" \
--cluster-ipv4-cidr "192.168.0.0/16" \
--machine-type "e2-medium" \
--num-nodes "1" \
--node-locations=asia-southeast1-a,asia-southeast1-b,asia-southeast1-c
```

Gambar 3. Setup *cluster* kubernetes menggunakan gcloud.

Setelah menginisiasi perintah pada gambar 3 hanya kurun waktu kurang lebih tiga menit, maka cluster kubernetes kita akan langsung tersedia dengan opsi yang telah kita berikan dimana versi kubernetes yang akan digunakan peneliti v1.17.15-gke.800 versi ini langsung di support secara langsung oleh tim google, dengan region asia-southeast1 menggunakan image Ubuntu 18.04 LTS sebagai sistem operasi node cluster google kubernetes engine, menggunakan flavor atau spesifikasi server 2 vCPU dan 4GB RAM, num-nodes berfungsi untuk menginisiasi server yang akan di deploy tiap zona, server akan dibuat di tiga zona yang berbeda yaitu *asia-southeast1-a*, *asia-southeast1-b* dan *asia-southeast1-c*. Kubernetes cluster dapat menangani struktur aplikasi microservices yang akan menjalankan banyak container, maka digunakan prefix /16. Setiap pods container akan memiliki alamat IP (Cluster IP). Jaringan overlay digunakan setiap pods container pada node yang berbeda yang menggunakan konsep tunneling vxlan dapat saling terhubung. Dalam Cluster kubernetes terdapat persistent volume yang berfungsi sebagai penyimpanan permanen dan sementara untuk pods dalam cluster. Persistent volume masing-masing digunakan pada pods database dan pods web application sebagai penyimpanan permanen dan sementara di tiap cluster google kubernetes engine. Pods mysql dan pods wordpress berkomunikasi menggunakan cluster IP. Kemudian aplikasi wordpress menggunakan type service Loadbalancer agar dapat diakses langsung melalui ip public dari cluster ke client.

C. Konfigurasi Aplikasi Web

Pengujian menggunakan apache web server dengan aplikasi wordpress dan MySQL 5.7. Gambar 4 merupakan tampilan dashboard website yang diuji. Tampilan pada gambar 4 merupakan salah satu contoh aplikasi wordpress yang sering digunakan sebagai blog, company profile maupun e-commerce.



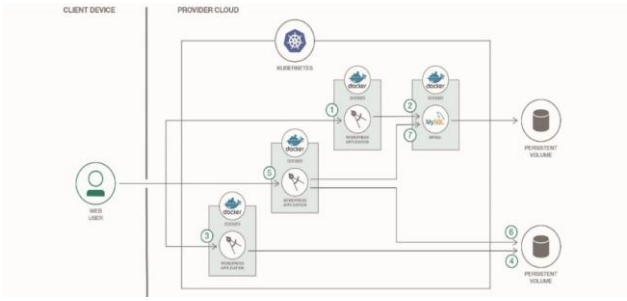
Gambar 4. Tampilan utama web.

Pengujian menggunakan wordpress sebagai aplikasi web yang paling banyak digunakan. Menurut data dari website managewp.com pada 28 april 2016, 23 Wordpress merupakan CMS (Content Management System)yang paling banyak digunakan dengan persentase 26.4% diseluruh dunia. Konfigurasi web dan database pada cluster google kubernetes engine berikut inisiasi konfigurasi web service wordpress dan database pada kubernetes yang di apply menggunakan manifest file yaml ditunjukan pada gambar 5.

```
gitclone https://github.com/ammarun11/k8s-wp.git
cd ~/k8s-wp
kubectl apply -f wordpressdb-deployment.yaml
kubectl apply -f wordpress-deployment.yaml
```

Gambar 5. Setup web dan database dengan manifest yaml.

Dalam Cluster kubernetes terdapat persistent volume yang berfungsi sebagai penyimpanan permanen untuk pods dalam cluster. Persistent volume masing-masing digunakan pada pods database dan pods web application sebagai penyimpanan permanen di tiap node worker. Pods MySQL dan pods wordpress berkomunikasi menggunakan cluster IP. Kemudian aplikasi wordpress menggunakan Loadbalancer agar dapat diakses langsung melalui IP public.



Gambar 6. Topologi aplikasi di dalam cluster google kubernetes engine

Di dalam cluster kubernetes gambar 6 menunjukan bagaimana user atau end-device dapat mengakses aplikasi web server, pada gambar tersebut menjelaskan dimana setiap node worker menggunakan docker sebagai container engine yang mendeploy aplikasi web service wordpress dan database engine mysql. untuk berkomunikasi dengan aplikasi didalam cluster kubernetes diperlukan perintah nodeport atau loadbalancer yang dapat diakses secara public.

HASIL DAN PEMBAHASAN

Nilai dari setiap parameter QoS didapatkan melalui 5 skenario pengujian dengan jumlah koneksi yang berbeda-beda yaitu 200, 500, 1000, 2000, dan 5000 koneksi. Request rate rata-rata setiap koneksi adalah sebesar 100 request per detik, dimana pengujian dilakukan berulang sebanyak 15 kali untuk mendapatkan hasil yang akurat serta mengurangi resiko kesalahan pengujian.

A. Proses Pengujian

Proses pengujian yang bertujuan untuk mendapatkan hasil dari kinerja aplikasi web *server*. Terdapat 5 skenario pengujian dengan jumlah koneksi yang berbeda-beda. Masing-masing skenario dilakukan uji coba sebanyak 15 kali, sehingga akan didapatkan data QoS rata-rata setiap parameter yang telah ditentukan. Pengujian beban dilakukan menggunakan *software* HTTPerf, kemudian *capture* jaringan menggunakan Wireshark untuk mendapatkan data yang akan diolah sesuai dengan parameter QoS. Jumlah koneksi, *request* per detik, dan banyaknya pengujian ditunjukkan pada Tabel 5.

Tabel 5. Skenario Pengujian

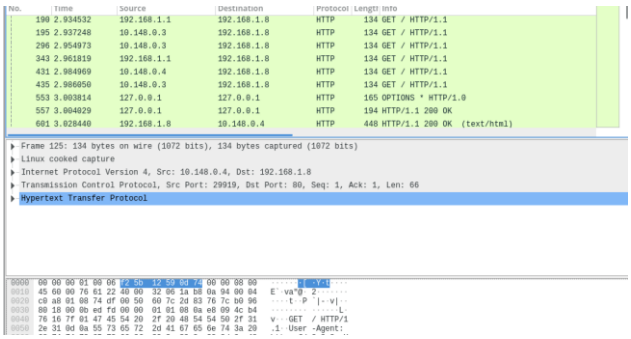
No	Jumlah Koneksi	*Request* per detik
1	100	100
2	250	100
3	500	100
4	1000	100
5	2000	100

Pengujian web server menggunakan stress tool web server HTTPerf dilakukan dengan memasukkan jumlah koneksi pada "*num-conns*" dan *request* per detik pada "*rate*", kemudian ditujukan ke alamat *Public* IP yang dibuat *Google Kubernetes Engine* seperti pada Gambar 7.

```
ammar@run:~$ httperf --server 35.198.239.45 --port 80 --rate 100 --num-conn 500 --num-call 1
```

Gambar 7. Pengujian dengan HTTPERF.

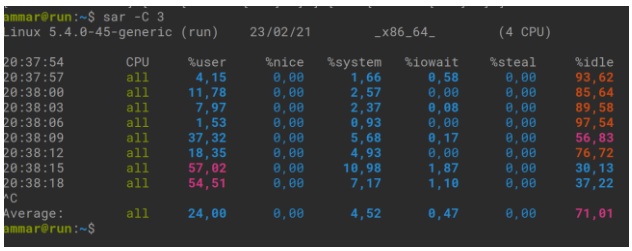
Paket data kemudian di-*capture* dan di-*filter* berdasarkan *transfer protocol* HTTP menggunakan Wireshark untuk diolah pada setiap pengukuran parameter QoS, seperti yang ditunjukkan Gambar 8.



Gambar 8. *Capture* Lalu Lintas Jaringan dengan Wireshark.

Pemantauan CPU Usage dilakukan pada setiap web server secara berkala setiap 2 detik menggunakan perintah "sar 2". Melalui perintah "sar", penulis mendapatkan informasi tentang rata-rata penggunaan CPU setiap server selama proses pengujian HTTPerf. Proses pemantauan ditunjukkan Gambar 9.

https://doi.org/10.30743/infotekjar.v5i2.3577



Gambar 9. Proses Pemantauan CPU *Usage.*

B. Pengukuran Nilai Availability

Availability diukur berdasarkan "nine", dimana semakin banyak "nine" maka semakin bagus tingkat ketersediaan suatu sistem. Pengujian ini dilakukan dengan mengamati apakah services web yang berjalan pada cluster tetap dapat berjalan saat terjadi kegagalan. Perhitungan availability berdasarkan rumus ditunjukan pada 1.

Tabel 6. Status Akses

Status			**Akses**
Node1	**Node2**	**Node3**	
hidup	hidup	mati	up
hidup	mati	hidup	up
mati	hidup	hidup	up

Tabel 6 menunjukan pada saat salah satu node atau dua node terjadi dimatikan aplikasi web tetap masih bisa diakses, karena fitur dari Google kubernetes engine yaitu auto-repair dimana fitur ini selalu menjaga kondisi tiap node pada cluster, Sehingga apabila terjadi insiden pada node dalam cluster google kubernetes engine maka secara otomatis akan melakukan perbaikan dengan sendirinya. Dan juga pods yang berada pada node yang mati otomatis akan tergantikan oleh pods yang sedang standby pada node yang tidak mengalami insiden. Sehingga aplikasi tetap dapat di akses.

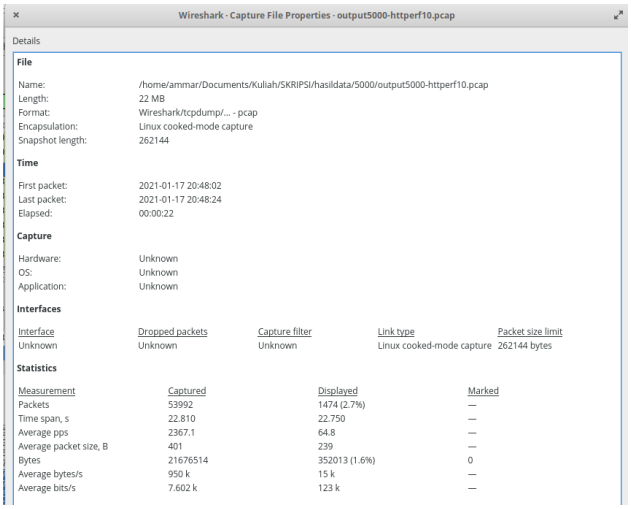
Tabel 7 Hasil Uji Coba *Availability*

Downtime		**Penyebab**	**Waktu untuk pulih**	**MTBF (menit)**	**MTTR (menit)**	**Availability (%)**
Tanggal	**Waktu**					
19 Januari 2021	23:12:29	node1 restart	23:12:56	1390	0.45	99.96
22 Januari 2021	10:51:40	node2 restart	10:52:01	2931	0.36	99.98
24 Januari 2021	23:57:18	node3 restart	23:01:16	7256	4.53	99.93
26 Januari 2021	21:23:00	node1 poweroff	21:23:09	8662	0.15	99.99
29 Januari 2021	21:40:43	node 3 poweroff	21:44:39	7287	4.27	99.94
Rata-rata *Availability*						99.96

Pengujian availability menggunakan skenario pada tabel 4.2 MTTR (Mean Time to Repair) merupakan berapa lama waktu yang diperlukan untuk mengembalikan layanan pada saat terjadi kegagalan pada cluster, MTBD (Mean Time Between Fail) merupakan parameter yang menunjukan berapa lama waktu uptime sebelum terjadinya insiden pada cluster. Dari total 27526 menit waktu uptime terdapat 9.76 menit yang digunakan untuk proses pemulihan web server agar dapat di akses kembali. Rata-rata availability pada saat pengujian sebesar 99.96%. Melalui percobaan pada tabel 4.2 waktu terlama untuk pemulihan web server agar dapat diakses kembali adalah pada saat melakukan restart atau power off pada node yang menjalankan pod Database MySQL. Karena sifat database yang statefull maka tidak mudah untuk melakukan replikasi pada pod database yang menyebabkan dia hanya ada pada node tertentu dan apabila node tersebut mati maka service web yang mengandalkan database tidak akan dapat diakses. Layanan kembali dan dapat diakses setelah 4.53 menit kemudian. Pada saat pengujian dengan melakukan restart dan power off pada salah satu node yang ada pada google kubernetes engine layanan web server masih bisa diakses karena, pods layanan web server dapat di replika ke node lain yang masih aktif, Sehingga waktu pemulihannya sangat cepat.

C. Pengukuran Nilai Throughput

Pengukuran throughput dilakukan untuk mengukur kemampuan sebenarnya jaringan dalam melakukan transmisi data. Nilai throughput didapatkan dari pembagian jumlah paket data yang diterima dengan waktu pengiriman data. Berdasarkan pengertian tersebut dapat disimpulkan bahwa semakin tinggi nilai throughput, maka semakin baik kualitas jaringannya. Wireshark telah 31 26 Januari 2021 21:23:00 node1 poweroff 21:23:09 8662 0.15 99.99 29 Januari 2021 21:40:43 node 3 poweroff 21:44:39 7287 4.27 99.94 Rata-rata Availability 99.96 memberikan nilai throughput dari paket data yang telah di-capture melalui menu Statistic seperti pada Gambar 10.

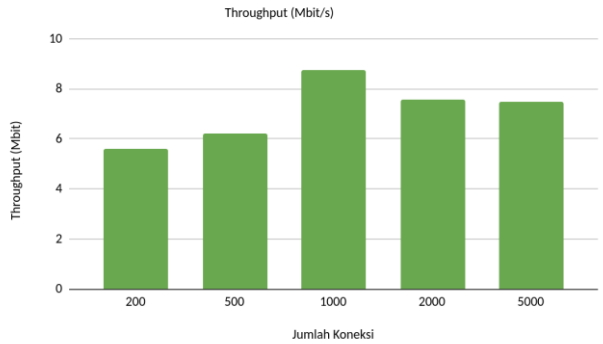


Gambar 10. Proses Pemantauan CPU *Usage.*

Nilai rata-rata throughput yang didapatkan pada setiap skenario pengujian ditunjukkan pada Tabel 8 dalam satuan Mbit/s dan grafik perubahan nilai throughput ditunjukkan pada Gambar 11.

Tabel 8. Nilai Rata-Rata *Throughput*

Parameter	Jumlah Koneksi	Hasil
Throughput (Mbit/s)	200	5.595
	500	6.216
	1000	8.764
	2000	7.581
	5000	7.473



Gambar 11. Grafik *Throughput*.

Pada 200 koneksi, didapatkan nilai rata-rata *throughput* sebesar 5.599 Mbit/s yang nilainya mulai meningkat pada 500 koneksi menjadi 6.216 Mbit/s dan kembali meningkat hingga koneksi 1000 dengan nilai *throughput* tertinggi sebesar 8.764 Mbit/s. Terjadi penurunan nilai *throughput* yang sangat kecil pada 2000 koneksi menjadi 7.581 Mbit/s dan mengalami penurunan nilai kembali pada 5000 koneksi menjadi sebesar 7.473 Mbit/s. Kenaikan nilai *throughput* terjadi karena penambahan jumlah koneksi yang secara otomatis meningkatkan jumlah *request* terhadap *web server* dalam waktu pengiriman yang pendek, sehingga terjadi peningkatan performa *web server*, namun ada efek dari sisi aplikasi web yang peneliti gunakan yaitu *wordpress* yang mengandalkan *threads* untuk menangani setiap user dalam melakukan *request*. aplikasi web tersebut akan mengalami penurunan koneksi jika *threads* yang tersedia sudah melebihi batas, efek ditunjukan pada 2000 dan 5000 koneksi yang mengalami penurunan.

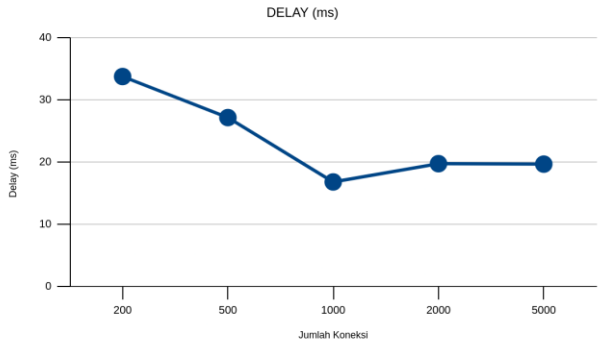
D. Pengukuran Nilai Delay

Pengukuran delay dilakukan untuk mendapatkan informasi waktu yang diperlukan paket hingga sampai ke tujuan selama proses transmisi. Rata-rata delay dihasilkan dari pembagian total delay keseluruhan dengan total paket yang diterima. Nilai-nilai tersebut terdapat pada menu Statistic Wireshark setelah 33 capture paket. Selanjutnya dilakukan perhitungan dan pengkategorian berdasarkan standar delay dari TIPHON, sehingga dihasilkan rata-rata delay yang ditunjukkan Tabel 9.

Tabel 9 Hasil Pengukuran Nilai Rata-Rata *Delay*

Parameter	Jumlah Koneksi	Hasil (ms)	Kategori
Delay	200	33.7238	Sangat Baik
	500	27.1261	Sangat Baik
	1000	16.7854	Sangat Baik
	2000	19.7222	Sangat Baik
	5000	19.6504	Sangat Baik

Delay merupakan variabel yang nilainya berubah-ubah, bergantung pada kondisi trafik. *Perubahan* nilai rata-rata *delay* pada setiap skenario pengujian ditunjukkan Gambar 12.



Gambar 10. Grafik Nilai Rata-Rata *Delay*

Pada 200 koneksi, didapatkan nilai delay sebesar 33.723 ms, kemudian trend rata-rata delay mengalami kenaikan hingga pada 1000 koneksi dengan Parameter Jumlah Koneksi Hasil (ms) Kategori Delay 200 33.7238 Sangat Baik 500 27.1261 Sangat Baik 1000 16.7854 Sangat Baik 2000 19.7222 Sangat Baik 5000 19.6504 Sangat Baik penambahan jumlah koneksi menjadi 27.126 ms pada 500 koneksi, 16.785 ms pada 1000 koneksi, Pada 2000 koneksi dan 5000 koneksi didapatkan delay 19.722 ms dan 19.650. Sempat menurun di koneksi 2000 dan 5000 dikarenakan ada faktor dari sisi aplikasi wordpress yang mengandalkan cache semakin banyak koneksi yang dikirim maka semakin cepat proses delay. Seluruh skenario menunjukkan kategori sangat baik, mengacu pada standarisasi TIPHON, yaitu kurang dari 150 ms. Kenaikan nilai rata-rata delay disebabkan oleh peningkatan jumlah koneksi. Berdasarkan nilai rata-rata delay yang dihasilkan, sistem masih dapat bekerja optimal hingga 5000 koneksi.

E. Pengukuran Nilai Packet Loss

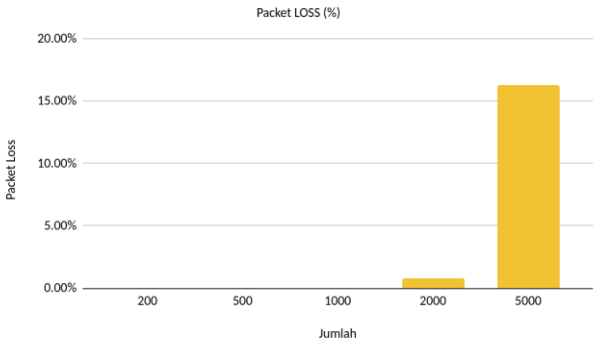
Pengukuran packet loss dilakukan untuk mendapatkan gambaran tentang jumlah total paket yang error selama proses transmisi. Presentasi rata-rata packet loss didapatkan dari hasil pembagian antara total error yang dihasilkan dengan jumlah request yang dikirim, kemudian dikonversi dalam bentuk persen. Total error didapatkan dari output HTTPerf, sedangkan jumlah request sama dengan jumlah koneksi, karena setiap koneksi mengirimkan 1 request. Nilai presentasi packet loss setiap skenario pengujian ditunjukkan Tabel 10 dan dikategorikan berdasarkan standarisasi dari TIPHON untuk mendapatkan nilai kelayakan packet loss.

https://doi.org/10.30743/infotekjar.v5i2.3577

Tabel 10. Hasil Pengukuran Nilai *Packet Loss*

Parameter	Jumlah Koneksi	Hasil	Kategori
Packet Loss (%)	200	0.00%	Sangat Baik
	500	0.00%	Sangat Baik
	1000	0.00%	Sangat Baik
	2000	0.76%	Sangat Baik
	5000	16.27%	Cukup Baik

Nilai kelayakan packet loss untuk skenario pengujian 200 hingga 2000 koneksi menunjukkan kategori sangat baik yang berarti sistem dapat melayani semua permintaan dengan minimum paket hilang selama proses transmisi, yaitu kurang dari 2%. Kemudian kategori cukup baik pada 5000 koneksi yang berarti presentasi packet loss ada di antara 15% hingga 25%, Berdasarkan persentase rata-rata packet loss, system bekerja sangat baik hingga 2000 koneksi, kemudian terjadi packet loss saat koneksi melebihi 2000 koneksi. Grafik perubahan presentasi packet loss ditunjukkan Gambar 11.



Gambar 11. Grafik Nilai Rata-Rata *Packet Loss*

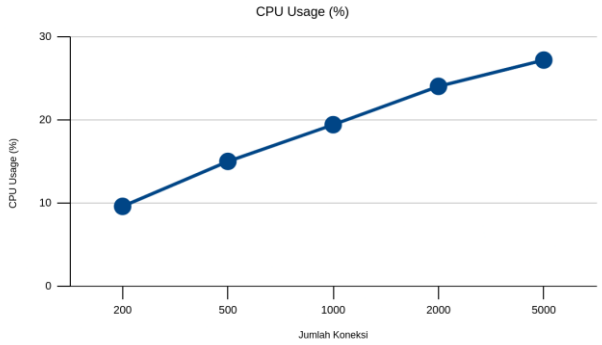
F. Pengukuran Nilai CPU Usage

Pengukuran CPU *usage* dilakukan untuk mendapatkan informasi tentang beban proses yang diterima setiap web server saat proses pengujian. Penggunaan 36 Parameter Jumlah Koneksi Hasil Kategori Packet Loss (%) 200 0.00% Sangat Baik 500 0.00% Sangat Baik 1000 0.00% Sangat Baik 2000 0.76% Sangat Baik 5000 16.27% Cukup Baik CPU dipantau pada masing-masing web server dengan keluaran nilai setiap 2 detik sekali. Rata-rata penggunaan CPU semua web server ditunjukkan Tabel 11.

Tabel 11. Hasil Pengukuran Nilai *CPU Usage*

Parameter	Jumlah Koneksi	Hasil
CPU *Usage* (%)	200	9.597%
	500	15.002%
	1000	19.411%
	2000	24.021%
	5000	27.178%

Presentasi rata-rata CPU usage seluruh web server pada koneksi 200 sebesar 9.597%, kemudian terus mengalami kenaikan seiring dengan bertambahnya jumlah koneksi, yaitu menjadi 15.002% pada 500 koneksi, 19.411% pada 1000 koneksi, 24.021% pada 2000 koneksi, dan 27.178% pada 5000 konesi. Trend kenaikan nilai rata-rata CPU usage tersebut divisualisasikan dalam bentuk grafik yang ditunjukkan Gambar 12.



Gambar 8. Grafik Nilai *CPU Usage*

Nilai CPU *usage* berbanding lurus dengan bertambahnya jumlah request atau koneksi yang diterima pada sisi cluster yang berarti bertambahnya pula beban request yang diterima oleh node selama proses pengujian. Berdasarkan 37 Parameter Jumlah Koneksi Hasil CPU Usage (%) 200 9.597% 500 15.002% 1000 19.411% 2000 24.021% 5000 27.178% nilai rata-rata pada Tabel 4.4, sistem masih dapat bekerja optimal hingga 5000 koneksi, hal ini ditandai dengan nilai rata-rata penggunaan CPU tertinggi masih jauh dari nilai puncak 100%, yaitu sebesar 27.178%.

PENUTUP

Berdasarkan hasil data dan analisis yang telah dilakukan, maka didapatkan kesimpulan yaitu, Dari hasil pengujian availability, didapatkan rata-rata availability 99.96% dan hasil pengujian sangat dipengaruhi oleh berapa lama durasi node untuk kembali normal ketika terjadi insiden pada cluster. Fitur auto-repair pada Google kubernetes engine membuat nodes yang mengalami insiden restart atau mati secara mendadak, akan kembali dihidupkan secara otomatis. Pada pengujian CPU usage load yang di dapat akan selalu meningkat dengan bertambahnya jumlah koneksi, yang berarti bertambah beban request yang diterima oleh cluster google kubernetes engine selama proses pengujian. Presentasi CPU usage pada setiap skenario pengujian yaitu 9.597%, 15.002%, 19.411%, 24.021%, dan 27.178%. Berdasarkan nilai rata-rata tersebut, sistem masih dapat bekerja secara normal dan optimal hingga 5000 koneksi, hal ini ditandai dengan nilai rata-rata penggunaan CPU tertinggi hanya sebesar 27.178%. Pada skenario pengujian throughput, Nilai rata-rata yang dihasilkan pada setiap konseki sebesar 7.126 Mbit/s dengan hasil pengujian throughput terbaik ketika diuji 1000 request ke cluster yaitu sebesar 8.754 Mbit/s. Kualitas service dari sisi delay menunjukan kualitas Sangat Baik menurut standar TIPHON yaitu kurang dari 150 ms. Hasil delay pada setiap skenario pengujian pada konseki 200, 500, 1000, 2000, 5000 yaitu 33.728 ms, 27.1256 ms, 16.785 ms, 19.722 ms dan 19.6504 ms. Presentasi packet loss berbanding lurus dengan kenaikan jumlah koneksi yang mengakses service web server. Semakin tinggi jumlah koneksi, maka semakin tinggi persentase paket yang hilang selama proses transmisi. Berdasarkan standarisasi TIPHON, pengujian dengan jumlah koneksi 200, 500, 1000 dan 2000 menunjukkan nilai kelayakan “Sangat Baik“ yaitu masing-masing sebesar

DAFTAR PUSTAKA

[1] International Telecommunication Union, "Measuring digital development Facts and figures 2019," ITUPublications, pp. 1–15, 2019.

[2] D. T. P. Kusuma, R. Munadi, and D. D. Sanjoyo, "Implementasi dan Analisis Computer Clustering System dengan Menggunakan Virtualisasi Docker," e-Proceeding Eng., vol. 4, no. 3, pp. 1–6, 2017.

[3] H. Shetty, S. Upadhaya, H. S. Rajarajeshwari, G. Shobha, and J. Chandra, "An empirical performance evaluation of docker container, openstack virtual machine and bare metal server," Indones. J. Electr. Eng. Comput. Sci., vol. 7, no. 1, pp. 205–213, 2017, doi: 10.11591/ijeecs.v7.i1.pp205-213.

[4] M. A. Nugroho and R. Kartadie, "Analisis Kinerja Penerapan Container untuk Load Balancing Web Server," JIPI (Jurnal Ilm. Penelit. dan Pembelajaran Inform., vol. 1, no. 02, pp. 7–15, 2016, doi: 10.29100/jipi.v1i02.35.

[5] M. A. Aprillio, "Three Way Handshake," p. 5, 2015

[6] Federico Culloca, "What is a web server?," Mozilla Developer Network, 2018. [Online]. Available: https://developer.mozilla.org/en-US/docs/Learn/Common_questions/What_is_a_web_server. [Accessed: 05-Apr-2020].

[7] A. Nugroho, W. Yahya, and Kasyful Amron, "Analisis Perbandingan Performa Algoritma Round Robin dan Least Connection untuk Load Balancing pada Software Defined Network," *J. Pengemb. Teknol. Inf. dan Ilmu Komput.*, vol. 1, pp. 1568–1577, 2017.

[8] D. Desmulyati and M. R. Perdana Putra, "Load Balance Design of Google Cloud Compute Engine VPS with Round Robin Method in PT. Lintas Data Indonesia," SinkrOn, vol. 3, no. 2, p. 147, 2019, doi: 10.33395/sinkron.v3i2.10064.

[9] Irfani and H. Sulistyanto, "Implementasi High Availability Server Dengan Teknik Failover Virtual Computer Cluster," 2015.

[10] ETSI, "Telecommunications and Internet Protocol Harmonization Over Networks (TIPHON); General aspects of Quality of Service (QoS)," *Etsi Tr 101 329 V2.1.1*, vol. 1, pp. 1–37, 1999.

 https://doi.org/10.30743/infotekjar.v5i2.3577