Desain dan Pengembangan Backend Aplikasi Bantucari Menggunakan *Microservices*

M Saif Alikhan[1], I Made Suartana[2]

[1,3] Jurusan Teknik Informatika, Fakultas Teknik, Universitas Negeri Surabaya
[1]msaifa.dev@gmail.com
[2]madesuartana@unesa.ac.id

***Abstrak*— BantuCari merupakan aplikasi untuk membantu seseorang apabila sedang kehilangan barang maupun menemukan barang. Aplikasi yang baik tentunya harus memiliki backend yang mumpuni. Salah satu cara untuk memnuhi backend mumpuni yakni dengan mengimplementasikan layanan *microservices*. Salah satu cara untuk membuktikan dengan pengujian *stress test* menggunakan aplikasi Apache JMeter dan dibandingkan dengan services lainnya seperti VPS dan Shared Hosting. Dengan menggunakan parameter response time, troughput, sent, receive dan error untuk mengetahui keunggulan *microservices*. Dan hasil yang didapatkan pada penelitian ini dengan penerapan *microservices* dan pengujian dengan skenario yang ditentukan, arsitektur *microservices* lebih unggul dibandingkan dengan layanan yang lainnya.**

***Kata Kunci*— Analisis dan Desain Sistem, Api Gateway, Apache JMeter, *Microservices*, *Stress test*.**

I. PENDAHULUAN

Saat ini perkembangan teknologi sudah sangat pesat oleh karena itu setiap orang dituntut untuk bisa lebih produktif. Sehingga akan membuat semua orang menjadi sibuk dengan aktivitas produktifnya. Akibat kegiatan yang semakin padat, membuat sebagian orang melupakan beberapa hal hingga berakibat seperti kehilangan barang yang dimiliki. Dilain sisi terdapat orang yang telah menemukan barang yang telah hilang, dengan niat mengembalikan barang tersebut namun terdapat kendala kesulitan untuk mengumumkan barang hilang tersebut. Meskipun pada zaman ini sudah terdapat teknologi untuk meyebarkan informasi seperti TV, Radio, Koran atau Media Sosial. Media social yang lagi trend belakangan ini bisa dijadikan alternatif sebagai sarana untuk memfasilitasi antara pemilik barang dan penemu barang hilang. Akan tetapi diperlukan suatu sistem dengan ketersediaan yang tinggi, karena interaksi antara pemilik dan penemu barang bisa sangat tinggi[1].

Beberapa teknologi bisa menjadi alternatif dalam membuat sistem dengan *availability* yang tinggi seperti Cloud Computing dan *Microservices*. Cloud Computing merupakan sebuah sistem komputasi parallel yang tersebar. Merupakan sekumpulan komputer yang terhubung secara virtual, yang ditampilkan sebagai satu atau lebih sumber daya yang dapat digunakan oleh pengguna melalui persetujuan antara penyedia jasa dan pengguna [2]. Secara umum, penggunaan Cloud Computing dalam aktivitas sehari-hari adalah dengan menggunakan aplikasi yang dapat diakses secara daring. Pengguna tidak perlu membeli perangkat lunak untuk melakukan pekerjaannya, karena sudah tersedia di internet. Penyedia jasa, dalam hal ini, menyediakan akses jaringan, perangkat lunak aplikasi, keamanan dan tempat penyimpanan data dalam bentuk pusat data yang terletak di internet[3]. Dengan adanya Cloud Computing, inovasi para pengembang semakin bertambah. Dengan latar belakang semakin bertambahnya trafik dan tempat untuk menampung data yang semakin bertambah. Sehingga dibutuhkan sebuah sistem yang memiliki fitur auto scalling. Hingga munculnya layanan baru bernama *microservices*. *Microservices* merupakan cara menyusun sebuah sistem perangkat lunak yang dibangun dari komponen independen kecil yang berinteraksi satu sama lain melalui jaringan seperti HTTP atau MQTT, layanan *microservices* menimbulkan peningkatan kompleksitas sistem karena seamkin rumitnya interaksi antar komponen individu [4]. Dibandingkan menggunakan sistem monolithic, arsitektur micoservices sendiri memiliki kelebihan, antara lain adalah : (a) Memiliki kompleksitas yang kecil dan sederhana, (b) Dapat mengembangkan aplikasi multi-platform, (c) Setiap service dapat berdiri sendiri, (d) Proses update hanya terjadi pada service yang akan dilakkan pembaruan [5]. Dengan menggunakan microservice, pengembang akan dimudahkan dalam melakukan penerapan aplikasi sehingga menjadikan produk atau hasil aplikasi yang efisien dan efektif. Dibuktikan dengan tidak perlunya melakukan merombak keseluruhan dari aplikasi karena telah terbagi menjadi komponen komponen yang kecil. Untuk mengembangkan *microservices* application dibutuhkan satu platform lagi yang dapat mendukung skema *microservices*. Yaitu docker, docker merupakan sebuah platform container yang digunakan untuk fungsi pengembangan, publikasi, hingga dapat digunakan oleh pengguna, dan proses yang dihasilkan menjadi lebih cepat dan mudah [6]. Dengan menggunakan container based virtulaization yang merupakan inti dari teknologi docker yang membuat lebih jauh efisien untuk mengembangkan aplikasi *microservices* [7].

Pada penelitian ini, akan di uji coba pendekatan arsitektur perangkat lunak yang berbeda untuk membuat sistem backend ”BantuCari”. Penelitian ini akan membandingkan arsitektur, Shared Hosting, VPS , dan Microservics untuk mencari performa terbaik pada kasus availability. Setelah menentukan *server* dengan performa terbaik, akan diimplementasikan backend yang akan digunakan untuk membantu dalam masalah kehilangan barang antara pihak penemu maupun pemilik barang. Layanan *microservices* dengan container based virtualization yang digunakan pada penlitian/percobaan ini adalah menggunakan lambda dari penyedia layanan Amazon Web Services (AWS). Dengan membadingkan tiga arsitektur tersebut akan didapatkan arsitektur dengan efisiensi dan efektifitas yang paling baik.¬

Sehingga sistem yang dihasilkan menjadi lebih cepat walau diakses dalam pengguna dengan skala yang besar.

II. METODOLOGI PENELITIAN

A. Instrumen Penelitian

Adapun spesifikasi dari *Hardware* (perangkat keras) dan spesifikasi *Software* (perangkat lunak) yang digunakan dalam melakukan pengujian yang dibutuhkan untuk sistem yang akan dibangun adalah sebagai berikut:

1) *Hardware* (Perangkat Keras), Spesifikasi *hardware* (Perangkat Keras) yang dibutuhkan sebagai *server* yang akan digunakan dalam penelitian seperti pada tabel 1

TABEL 1
KEBUTUHAN PERANGKAT KERAS DAN SPESIFIKASINYA

Hardware	**Spesifikasi *hardware***
VPS	CPU : 1 Core RAM : 1 GB Storage : 40 GB
Shared Hosting	CPU : 1.5 Core RAM : 2 Core Storage : Unlimited
Microservices	RAM : 1024 MB Storage : 512 MB (Sementara)
Database *Server*	CPU : 1 Core RAM : 1 GB Storage : 40 GB

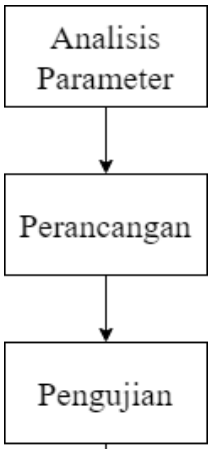
2) *Software* (Perangkat Lunak), yang digunakan dalam perancangan sistem dan pengujian kinerja dalam penelitian ini seperti pada tabel 2.

TABEL 2
KEBUTUHAN PERANGKAT LUNAK

Software	**Deskripsi**
Apache JMeter	Digunakan untuk melakukan proses *stress test*ing pada penelitian ini
Termius	Digunakan untuk melakukan remote pada *server* VPS yang akan dilakukan pengujian.
Chrome	Digunakan untuk melakukan manajemen *server* VPS, Shared Hosting dan Lambda

B. Prosedur Penelitian

Pada gambar 1 merupakan gambaran dari prosedur penelitian yang dimilai dari proses pengumpulan data untuk mendapatkan data yang benar dan meyakinkan agar hasil dari penelitian dapat berjalan sesuai dengan tujuan yang sudah diterapkan sebelumnya, penulis melakukan langkah-langkah sebagai berikut:



Gbr 1 Prosedur Penelitian

1) *Analisis*, Tahap ini akan digunakan untuk merencanakan proses pengiriman data dari *server* ke *client*. Dalam tahap ini akan dirancang sebuah proses untuk mengirimkan data dari *server* ke *client*, sehingga dapat diukur parameter-parameter seperti waktu tanggapan, *throughput*, penggunaan *RAM*, penggunaan *CPU*, data yang dikirim, data yang diterima dan jumlah error. Kemudian akan diambil kesimpulan dari hasil perbandingan yang telah diperoleh..

2) *Perancangan*, *Backend* akan digunakan dalam penerapan aplikasi bantuCari, namun sebelum dilakukan penerapan pada aplikasi. Akan dilakukan pengujian yang akan dibahas pada penelitian kali ini. *Backend* yang dibuat kali ini menggunakan Bahasa pemrograman javascript dan dimodelkan menjadi *REST API* yang akan dipanggil oleh aplikasi. Kemudian data yang telah diciptakan oleh pengguna akan disimpan pada database *MySQL*. Berikut tahapan untuk melakukan perancangan *Backend*:

- Analisa Kebutuhan Fungsional Analisa Kebutuhan Fungsional merupakan tahap awal dalam menentukan kebutuhan-kebutuhan fungsional yang akan dibutuhkan oleh sistem. Kebutuhan tersebut yang nantinya akan diterapkan pada aplikasi BantuCari. Kebutuhan fungsional terdiri dari kebutuhan aktor dan kebutuhan fungsional. Aktor merupakan pengguna yang akan menggunakan aplikasi. Sedangkan kebutuhan fungsional adalah fungsi dari aplikasi. Berikut beberapa daftar kebutuhan fungsional pada aplikasi dapat dilihat pada tabel 3.

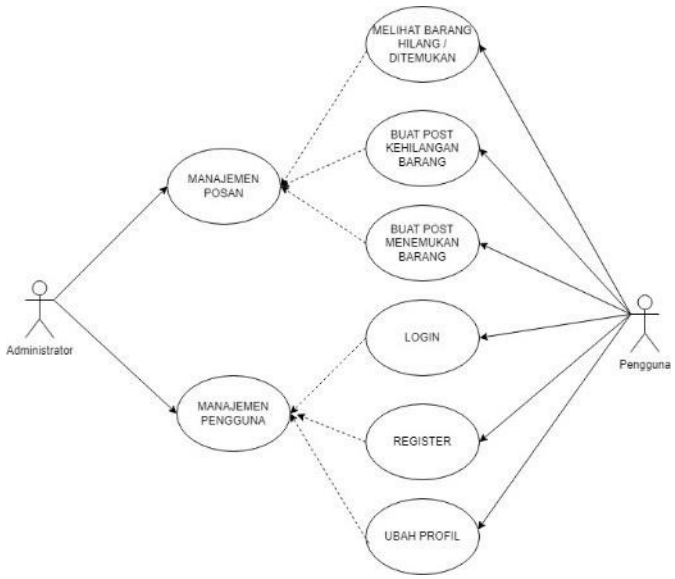
TABEL 3
KEBUTUHAN FUNGSIONAL

ID	Aktor	Kebutuhan fungsional
AD01	Administrator	Manajemen Pengguna
AD02	Administrator	Manajemen Posan
US01	Pengguna	Registrasi Pengguna
US02	Pengguna	Manajemen Data Posan

Dari tabel 1 merupakan kebutuhan fungsional untuk pembuatan *microservices*. Terdapat 4 item, yang pertama adalah manajemen pengguna yang dilakukan oleh aktor administrator, kedua manajemen posan dengan aktor administrator, ketiga Registrasi Pengguna oleh aktor *public user*, dan terakhir manajemen data posan dengan aktor *public user*.

- Perancangan Perangkat Lunak

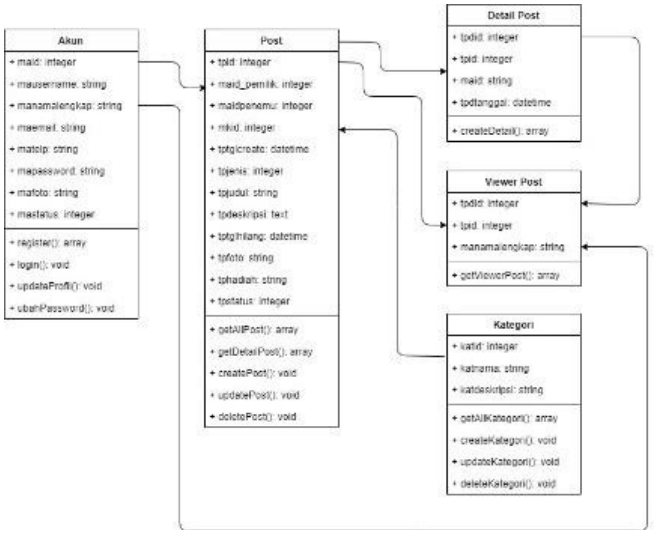
Tahap selanjutnya melakukan perancangan terhadap arsitektur dari perangkat lunak. Perancangan kali ini dibuat menggunakan metode use case. yang ada pada gambar 2:



Gbr 2 Use Case Aplikasi BantuCari

Pada gambar 2 dijelaskan terdapat 8 action. Dan terdapat 2 aktor antara lain ialah aktor administrator dan aktor pengguna. Pada aktor pengguna dapat melakukan aktivitas Login, Register, dan Ubah Profil yang dimana dapat dipantau oleh Administrator melalui aktivitas manajemen posan. Kemudian pengguna juga dapat melakukan tambah posan baru dan melihat posan yang juga dapat dipantau oleh administrator melalui aktivitas Manajemen Pesan.

- Membuat Desain Database Membuat tabel yang saling terhubung antara satu tabel dengan tabel yang lain. Tabel yang akan dibuat antara lain tabel akun digunakan untuk kumpulan data akun, kategori digunakan untuk mengumpulkan data kategori dari barang yang hilang, tabel post digunakan untuk data pengguna yang kehilangan maupun menemukan barang yang hilang, tabel post_detail untuk mencatat pengguna yang melihat data post dan tabel viewer_post sebagai tabel yang digunakan untuk mengumpulkan data penonton dari post yang sudah ada . Untuk relasi dapat dilihat pada gambar 3.



Gbr 3 Class Diagram

- Pemodelan *Microservices* Setelah menganalisa kebutuhan fungsional. Tahap selanjutnya adalah membuat model dari *microservices*. Tahap ini akan mengkategorikan setiap fungsi berdasarkan kebutuhan dan berdasarkan data yang akan diolah. Dimana dalam satu kategori akan berisi satu fungsi atau lebih. Dalam kata lain, *microservices* merupakan group fungsi yang terdiri satu atau lebih layanan didalamnya. Dan mereka saling terhubung antara satu dengan yang lain.

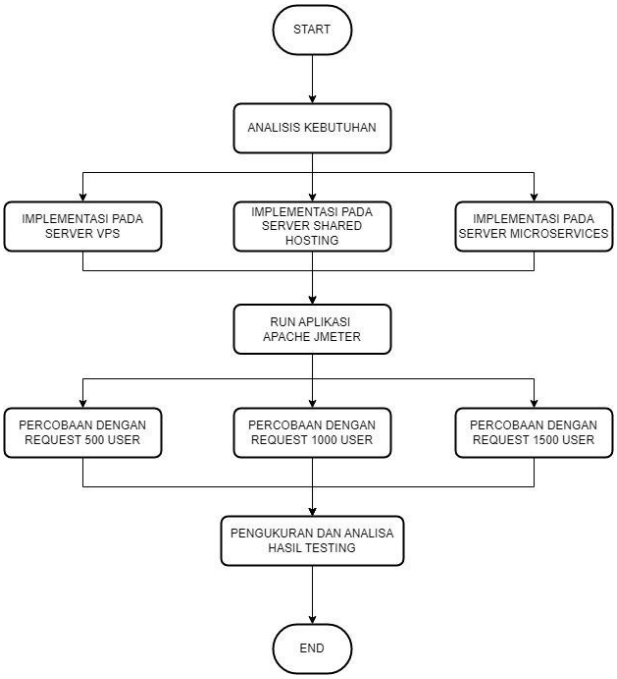
TABEL 4
KEBUTUHAN *MICROSERVICES*

ID	Kebutuhan *microservices*
AD01	Login aplikasi
AD01	Blokir Pengguna
AD02	Mengambil Data Semua Posan
US01	Login Aplikasi
US01	Registrasi Aplikasi
US02	Mengambil data posan yang dimiliki
US02	Mengambil data detail posan
US02	Membuat posan baru
US02	Mengubah posan yang sudah ada
US02	Menghapus posan yang sudah ada

C. Skenario Pengujian

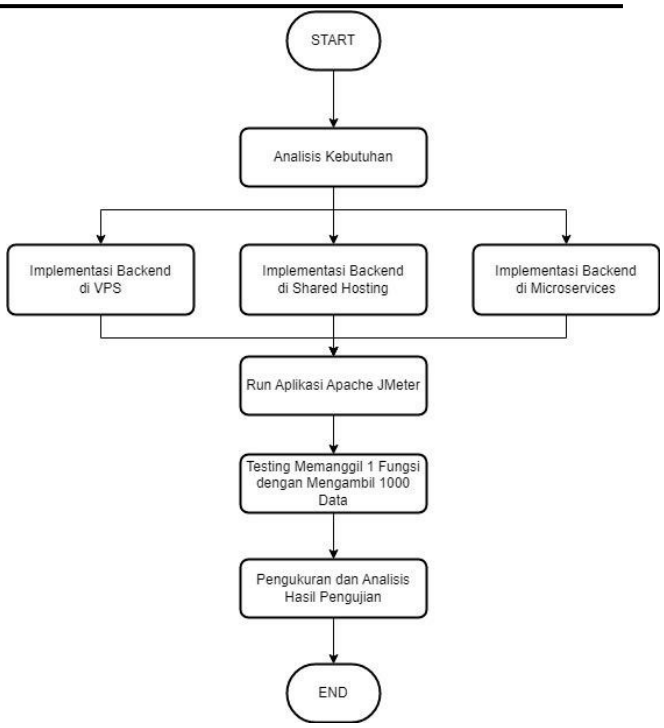
Analisis perbandingan performa *web server Virtual Private Server, Shared Hosting*, dan *Microservices* akan dilakukan menggunakan software Apache JMeter dengan metode uji beban [8]. Proses pengujian kinerja ini akan dilakukan melalui beberapa tahap yang akan dijadikan sebagai prosedur penelitian oleh penulis. Adapun tahapan-tahapan prosedur penelitian yang akan diujikan dapat dilihat dibawah ini:

1) *Skema Rancangan Pengujian*, Pada Gambar 4 Pengukuran dilakukan dengan mengambil nilai rata-rata dari masing-masing pengujian yang dilakukan pada *backend* dengan beban yang diberikan yaitu 500, 1000 dan 1500 *request*[9].



Gbr 4 Skema Tahap Penelitian

2) *Skema Rancangan Alur Pengujian Web Server* Gambar 5 menjelaskan alur skema pengujian web *server*. Tahapan-tahapannya adalah : Setelah proses booting dari sistem operasi selesai dilakukan, tahap selanjutnya adalah instalasi web *server* dan perangkat lunak yang diperlukan. Kemudian, tahap selanjutnya adalah instalasi Apache JMeter pada komputer *client* dan melakukan uji beban. Setelah itu, tahap selanjutnya adalah menjalankan salah satu fungsi dari masing-masing web *server* yang akan diuji. Tahap terakhir adalah menarik kesimpulan dari semua hasil pengujian untuk membandingkan performa dari masing-masing web *server* yang diuji.



Gbr 5 Skema Pengujian Web *Server*

D. Parameter kinerja

Parameter kinerja adalah proses penilaian terhadap pengukuran performa suatu sistem atau aktivitas berdasarkan tujuan dan sasaran yang telah ditentukan sebelumnya. Hasil dari proses ini dibandingkan dengan tujuan yang ingin dicapai dan efektivitas tindakan yang diambil dalam mencapai tujuan tersebut. Parameter yang digunakan dalam pengujian meliputi waktu tanggapan, *throughput*, penggunaan RAM, penggunaan CPU, data yang dikirim, data yang diterima, jumlah error, dan log.. Berikut penjelasan dari parameter yang sudah disebutkan diatas adalah:

1) *Response Time* adalah waktu yang dibutuhkan oleh sebuah antarmuka untuk merespon permintaan dari pengguna. Ini adalah perbedaan waktu antara saat permintaan dikirim dan saat respon diterima sepenuhnya.

2) *Throughput* adalah kecepatan yang efektif dari transfer data yang diukur dalam bps (bit per second). *Throughput* adalah jumlah total paket yang diterima dan diamati pada tujuan selama periode waktu tertentu, kemudian dibagi dengan durasi periode waktu tersebut.

3) *RAM USAGE*, merupakan pemakaian ruang penyimpanan sementara pada *server*. Semakin besar jumlah RAM yang tersedia, maka website akan semakin cepat diakses, dapat memproses lebih banyak permintaan dari pengunjung, dan dapat menangani traffic yang intens dengan baik.

4) *CPU USAGE*, adalah inti utama untuk menerima perintah dari pengguna website dan melakukan tindakan

sesuai dengan perintah tersebut. Semakin banyak jumlah core pada *server*, maka akan semakin cepat pula perintah yang dikerjakan.

5) *Sent* merupakan jumlah data yang dikirimkan dari *server* ke *client* saat pengujian dilakukan. Semakin kecil nilainya, maka hasilnya akan semakin baik.

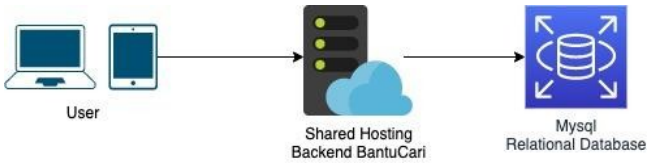
6) *Received* adalah jumlah data yang diterima dari *client* menuju *server* saat pengujian dilakukan. Semakin kecil nilainya, maka hasilnya akan semakin baik.

7) *Error* adalah kesalahan yang terjadi saat *client* mengakses halaman web atau saat *server* mengirimkan balasan ke *client*. Semakin banyak error yang terjadi pada sebuah *server*, maka akan semakin buruk pula kinerja dari web *server* tersebut.

III. HASIL DAN PEMBAHASAN

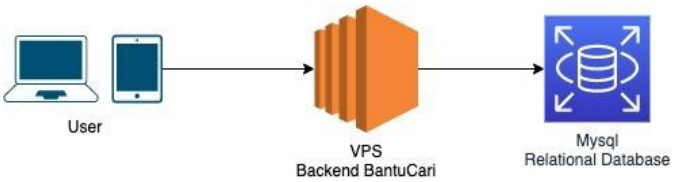
A. *Arsitektur Server*

Sebelum melakukan pengujian dari ketiga *server*. Hendaknya melakukan perancangan arsitektur yang akan dibuat. Berikut merupakan hasil dari perancangan arsitektur dari masing masing *server*



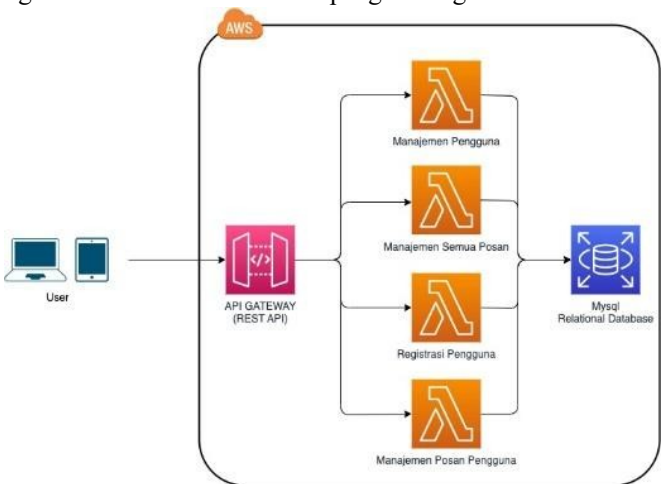
Gbr 6 Arsitetur Shared Hosting

Pada gambar 6 merupakan arsitektur dari *Shared Hosting* dimana pada arsitektur tersebut memiliki 3 perangkat penting, pertama ialah pengguna atau aplikasi yang akan mengakses, kedua ialah *server* shared hosting yang digunakan sebagai tempat berjalannya backend dari aplikasi, ketiga adalah *server* mysql yang digunakan untuk database dari aplikasi.



Gbr 7 Arsitektur VPS

Pada gambar 7 merupakan arsitektur jika menggunakan *server* VPS. Pada arsitektur VPS terdapat 3 komponen penting yang sama seperti pada *server* shared hosting. Namun, perbedaanya pada *server* backend. Pada VPS *server* nya bersifat *private,* atau dalam satu perangkat computer digunakan sendiri, berbeda dengan *shared hosting* yang digunakan Bersama sama oleh pengembang lain.

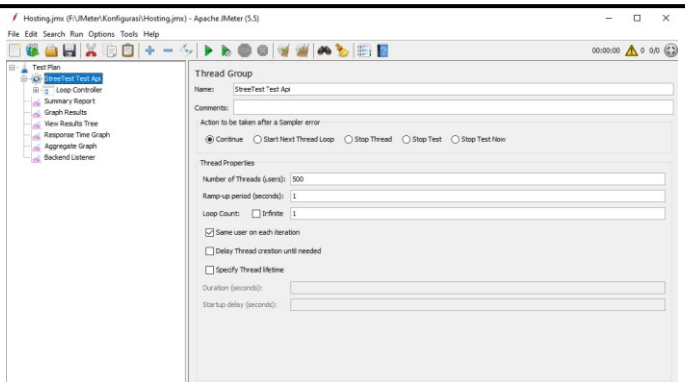


Gbr 8 Arsitektur *Microservices*

Dari gambar 8 merupakan arsitektur dari *microservices.* berbeda dengan *server VPS* atau *Shared Hosting, Microservices* memiliki komponen yang lebih banyak. Pertama adalah *API GATEWAY* [10] yang digunakan sebagai jembatan antara pengguna aplikasi dan *microservices,* kemudian pada *server microservices* memilki jumlah yang banyak sesuai dengan jumlah kategori fungsi yang dibutuhkan. Sehingga dalam *microservices* fungsi satu dengan yang lainnya tidak akan mengganggu jika terdapat kendala atau *error*.

B. *Konfigurasi Apache JMeter*

Dalam aplikasi Apache JMeter, akan dilakukan uji beban (*stress test*) untuk mengetahui nilai dari *throughput*, waktu tanggapan, data yang dikirim, data yang diterima dan error dengan melakukan konfigurasi jumlah pengguna dan waktu yang dibutuhkan pengguna untuk mengakses sistem backend.. Pada penelitian kali ini, penulis akan melakukan pengujian pada masing masing *server* yang telah dibuat. Dengan konfigurasi simulasi jumlah pengguna yang melakukan request secara bersamaan sebanyak 500, 1000 dan 1500 masing masing diakses dalam waktu 1 detik dan 5 detik. Sehingga akan terjadi 6 tahap proses untuk masing masing *server*. Pada gambar 9 merupakan contoh konfigurasi yang akan digunakan dalam pengujian kali ini

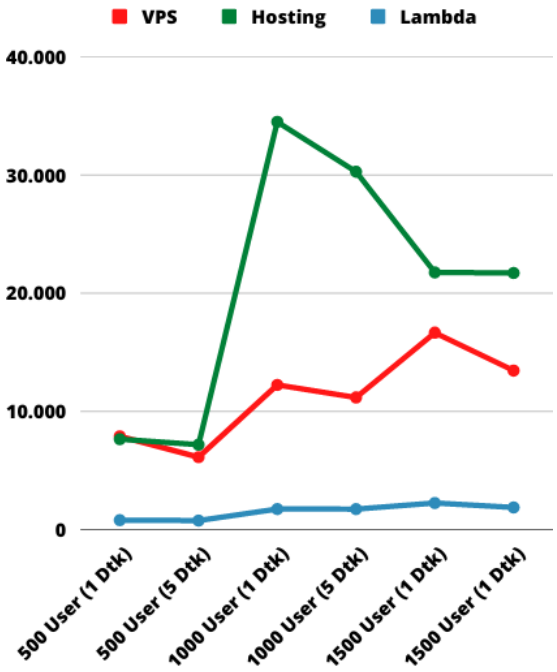


Gbr 9 Konfigurasi Pengujian 500 user dalam 1 detik

C. *Pengujian Backend*

Pengujian kinerja *server* akan dilakukan secara bergantian dengan diawali oleh pengujian terhadap shared hosting, disusul dengan VPS dan terakhir Lambda. Dengan menggunakan metode *stress testing*. Pengujian akan dilakukan dengan menjalankan perintah fungsi yang sama hanya berbeda penerapan *server*.

Pengujian dilakukan dengan cara memanggil fungsi untuk mengambil data dari database dengan query yang terdapat sambungan dengan beberapa tabel dan mengamibl 1000 data. Berikut merupakan hasil dari *stress testing* pada gambar 10 ini:



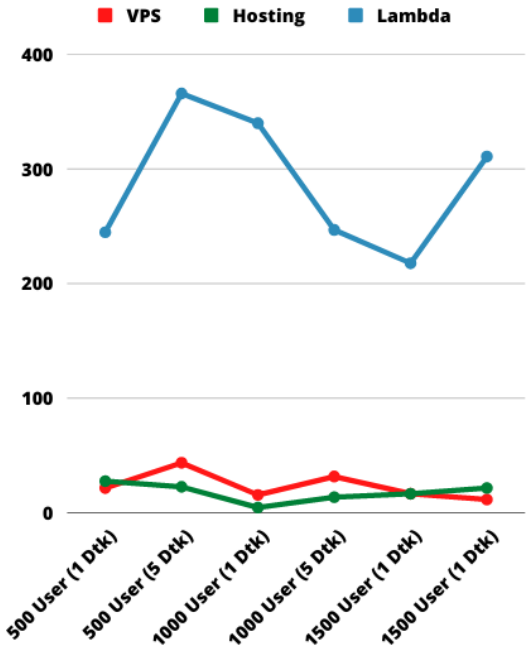
Gbr 10 Hasil Pengujian dalam response time

Dari gambar 10 hasil dari pengujian *response time* pada *server* VPS memiliki hasil yang cenderung meningkat. Pada *server* ini dihasilkan pola pengujian dengan waktu 1 detik akan lebih tinggi hasilnya disbanding dengan pengujian dengan waktu 5 detik. Pola tersebut didapatkan dari pengujian dengan 500, 1000, dan 1500 user.

Dari gambar 10 hasil dari pengujian response time pada *server* shared hosting memiliki hasil yang sama pada pengujian 500 user dalam waktu 1 maupun 5 detik. Kemudian melambung jauh ketika dilakukan pengujian 1000 user dengan waktu 1 detik, dan turun ketika waktu pengujian dilakukan selama 5 detik. Kemudian pada pengujian 1500 user, *server* shared hosting juga memiliki nilai yang stabil ketika diakses dalam waktu 1 atau 5 detik.

Pada gambar 10 hasil dari pengujian *response time* pada *server microservices* memiliki nilai yang cenderung stabil. Dari pengujian 500, 1000, dan 1500 user dengan jangka waktu 1 atau 5 detik. *Server microservices* mendapat nilai yang stabil dengan di bawah 1000 ms.

Setelah mendapatkan hasil dari pengujian, maka dapat disimpulkan bahwa untuk reponse time yang paling cepat terdapat pada *server microservices*. Dengan hasil yang stabil dari 500 pengguna maupun 1500 pengguna. Kemudian pada gambar 10 merupakan hasil dari proses *troughput* dari masing masing *server*



Gbr 11 Hasil pengujian dalam troughput

Pada gambar 11 merupakan hasil pengujian *troughput* dari *server* VPS, dimana mendapatkan hasil stabil dari pengujian 500, 1000, dan 1500 user. Perbedaan dari setiap Langkah hanya berselisih sekitar 20 poin.

Pada gambar 11 merupakan hasil pengujian *troughput* dari *server Shared Hosting* mendapatkan hasil yang

cenderung menurun dari pengujian 500, 1000, dan 1500 user. Dari hasil tersebut, didapatkan pola ketika pengujian dengan waktu 5 detik. Didapatkan hasil yang lebih tinggi disbanding 1 detik.

Pada gambar 11 merupakan hasil pengujian troughput dari *server Microservices* mendapatkan hasil yang tinggi, namun tidak teratur. Karena pada pengujian 500 user, hasil yang didapatkan ialah peningkatan. Kemudian dengan pengujian 1000 user hasil yang didapatkan ialah penurunan. Dan terakhir pengujian dengan 1500 user kembali terjadi peningkatan.

Dengan melihat gambar 11 dapat disimpulkan bahwa untuk troughput terbaik ialah pada *server microservices*, dengan hasil yang jauh lebih tinggi dari pada *server* yang lain. Untuk *server* VPS dan shared hosting sama sama mendapatkan hasil dibawah 100 per second. Selain data grafik antara *response time* dan *troughput,* terdapat hasil lainnya seperti error, troughput, received dan sent. Untuk *server* VPS ada pada tabel 5.

TABEL 5
HASIL PENGUJIAN PADA VPS

	Error	Troughput	Received (KB/Sec)	Sent (KB/Sec)
500 User (1 Detik)	3.93%	22	916	4.56
500 User (5 Detik)	4.43%	45	1844	9.18
1000 User (1 Detik)	16.19%	19.7	715.56	3.54
1000 User (5 Detik)	3.76%	32.8	1357.96	6.76
1500 User (1 Detik)	19.67%	17.6	612.8	3.02
1500 User (5 Detik)	12.85%	46.8	1760.53	8.72

Dari tabel 5, dapat diketahui jika menggunakan *server VPS* maka yang didapatkan adalah Error yang terdapat hingga dua kali lipat, *troughput* yang meningkat jika testing dilakukan dengan durasi 5 detik. Parameter *Received* dan *Sent* yang bisa dua kali lipat antara durasi 1 detik dan 5 detik.

Untuk hasil pengujian dari *server* shared hosting dapat disajikan pada tabel 6 :

TABEL 6
HASIL PENGUJIAN PADA SHARED HOSTING

	Error	Troughput	Received (KB/Sec)	Sent (KB/Sec)
500 User (1 Detik)	42.04%	28.2	733.77	3.48
500 User (5 Detik)	3.30%	23.1	994.14	4.92
1000 User (1 Detik)	1.80%	14.7	621.34	3.07
1000 User (5 Detik)	1.60%	17	719.92	3.55
1500 User (1 Detik)	23.87%	17.6	590.81	2.86
1500 User (5 Detik)	20.07%	22.6	793.22	3.85

Dari tabel 6 dapat diartikan bahwa menggunakan *server shared hosting* akan mendapatkan hasil yang kurang stabil, dikarenakan arsitektur shared hosting ialah berbagi dengan pengguna lain sehingga kurang stabil dalam performanya. Namun pada parameter *Received* dan *Sent* mendapat hasil yang cenderung stabil.

Dan terakhir untuk hasil pengujian dari *microservices* lambda juga disajikan dalam tabel 7.

TABEL 7
HASIL PENGUJIAN PADA MICORSERVICES

	Error	Troughput	Received (KB/Sec)	Sent (KB/Sec)
500 User (1 Detik)	90.35%	245.9	1084.18	60.99
500 User (5 Detik)	89.47%	366.9	1760.82	91.01
1000 User (1 Detik)	88.04%	340.6	1843.46	84.5
1000 User (5 Detik)	88.26%	247.9	1318.81	61.49
1500 User (1 Detik)	89.16%	218.3	1077	54.15
1500 User (5 Detik)	90.63%	311	1340	77.13

Dari tabel 7 dapat diartikan bahwa jika menggunakan *server microservices* hasil didapatkan cenderung stabil. Antara hasil *Error, Troughput, Received*, maupun *Sent* memiliki hasil yang tidak berbeda jauh.

Dari hasil penelitian diatas dapat disimpulkan bahwa *server* yang paling unggul ialah *microservices*, karena unggul dalam aspek *response time, troughput, received,* dan *sent* dibandingkan dengan *server* VPS maupun shared hosting. *Server microservices* juga memiliki hasil testing yang stabil, dibanding dengan *server* lainnya.

IV. KESIMPULAN

Berdasarkan hasil analisa dari pembahasan yang sudah dilakukan sebelumnya, maka dalam pembuatan *microservices* diperlukan beberapa layanan lain seperti Api Gateway dan Database agar menjadi suatu layanan yang utuh dan bisa digunakan sepenuhnya oleh aplikasi.

Berdasarkan hasil Analisa perbandingan antara *server* VPS, Shared Hosting dan *Microservices*. *Server* yang paling tepat untuk terapkan pada backend aplikasi BantuCari ialah menggunakan *microservices.* Dikarenakan hasil dari *stress testing microservices* memiliki hasil yang lebih stabil dibandingkan dengan *server* VPS maupun Shared Hosting. Dibuktikan dengan pengujian menggunakan metode *stress testing*, *server microservices* ketika diakses oleh 500 maupun 1500 user secara bersamaan memiliki hasil yang tidak berbeda jauh.

Pengembangan dari penelitian ini dapat dilakukan dalam pengujian dengan skenario yang lain dan menggunakan

metode testing yang lainnya agar hasil yang didapatkan bisa lebih baik dan dapat digunakan untuk penelitian seterusnya.

REFERENSI

[1] Latif, Y. (2016). *SISTEM PENCARIAN DAN PENGUMUMAN BARANG HILANG.* Makassar: FAKULTAS SAINS DAN TEKNOLOGI UIN ALAUDDIN MAKASSAR.

[2] L, G. W. (2011). Optimized Management of Power and Performance for Virtualized Heterogeneous *Server* Clusters. *IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, 23-32.

[3] Katzan. (2010). On the privacy of cloud computing. *International Journal of Management and Information Systems*, 1-12.

[4] Suryotrisongko, H. (2017). "Arsitektur Microservice untuk Resiliensi Sistem Informasi. *OAJIS: Open Access Journal of Information System vol. 6*, 235-250.

[5] Rafiqi, D. (2019). IMPLEMENTASI ARSITEKTUR MICROSERVICE PADA APLIKASI ONLINE TRAVEL TOURINC. *KARYA ILMIAH MAHASISWA MANAJEMEN INFORMATIKA*, 1-9.

[6] Fihri, M. (2019). Implementasi & Analisis Performansi Layanan Web Pada Platform Berbasis Docker. *e-Proceeding of Engineering*, 39964001.

[7] Eko, S. (2021). Analisa dan Implementasi Microservice pada Container Menggunakan Docker. *CoMBInES: Conference on Management Business vol 1*, 557-564.

[8] Sonia Ginasari, N., Suar Wibawa, K., & Ayu Wirdiani, N. (2021). Pengujian *Stress test*ing API Sistem Pelayanan dengan Apache JMeter. JITTER : Jurnal Ilmiah Teknologi Dan Komputer, 2(3), 552-557. Retrieved from https://ojs.unud.ac.id/index.php/jitter/article/view/79652.

[9] Prasetyo, S., & Wijaya, A. (2021). Analisa dan Implementasi Microservice pada Container Menggunakan Docker. CoMBInES - Conference On Management, Business, Innovation, Education And Social Sciences, 1(1), 557-564. Retrieved from https://journal.uib.ac.id/index.php/combines/article/view/4480.

[10] Polladino, Marco. (2017). *Microservices* & API Gateways, Part 1: Why an API Gateway ?.Dipetik November 4, 2017 dari https://www.nginx.com/blog/*microservices*-api-gateways-part-1-why-an-apigateway/.