Tugas Mandiri 2
LAW 2022
Performance Testing

Muhammad Alif Saddid
1906398250
LAW - A

Webservice yang Diuji

Webservice yang diuji adalah http://host-1906398250-port-58250.proxy.infralabs.cs.ui.ac.id/
Dengan endpoint-endpoint berikut:

- **GET** http://host-1906398250-port-58250.proxy.infralabs.cs.ui.ac.id/roles (mendapatkan semua roles)
- **POST** http://host-1906398250-port-58250.proxy.infralabs.cs.ui.ac.id/roles (membuat role baru)
- **PUT** http://host-1906398250-port-58250.proxy.infralabs.cs.ui.ac.id/roles/:id (mengupdate sebuah role)

Swagger dapat diakses melalui
http://host-1906398250-port-58250.proxy.infralabs.cs.ui.ac.id/swagger/index.html

Alasan Pengujian

Endpoint roles tersebut merupakan endpoint yang **saya buat** untuk PPL. Endpoint-endpoint tersebut akan sangat sering dipanggil untuk memeriksa apakah seorang user berhak mengakses laman tertentu / endpoint tertentu. Oleh karena itu, perlu dilakukan performance testing pada endpoint-endpoint tersebut untuk memastikan pemanggilan endpoint tersebut tidak mengurangi kenyamanan pengguna dalam menggunakan aplikasi.

Skenario yang Diterapkan

Dilakukan pengaksesan 3 endpoint tersebut, dengan skenario:

- 0.01% POST
- 0.99% PUT
- 99% GET

Skenario tersebut dipilih karena biasanya aktivitas membuat role (POST) dan mengupdate role (PUT) jauh lebih sedikit dibandingkan pemanggilan endpoint untuk mendapatkan data role (GET).

Software yang Digunakan

1. k6 untuk mengirim request
2. InfluxDB untuk menyimpan data metrics
3. Grafana untuk menampilkan data metrics hasil pengujian
4. Telegraf untuk mencatat data metrics CPU utilization
5. Nginx sebagai reverse proxy supaya webservice bisa diakses tanpa vpn

Persiapan

1. Menginstall influxdb, telegraf, dan nginx di container infralabs
2. Membuat sebuah compute engine GCP, menginstall docker dan nginx.
3. Setup nginx di infralabs agar webservice bisa diakses tanpa vpn
4. Membuat skenario performance test menggunakan k6

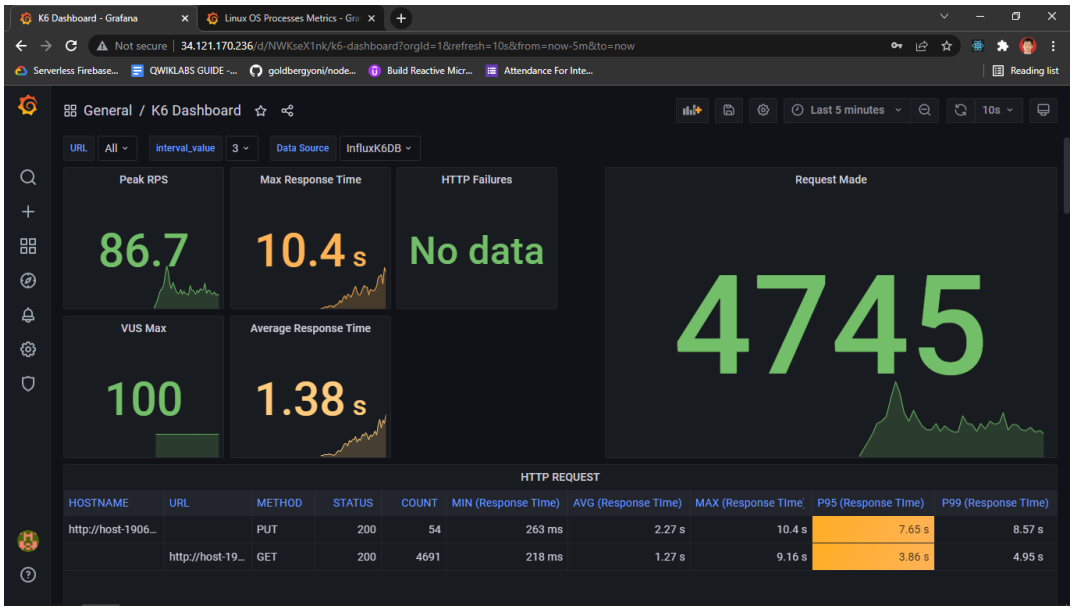
```
export default function () {
    // default header parameter
    const params = {
        tags: {
            name: '',
            host: `${__ENV.API_HOSTNAME}`
        },
        name: randomString(5),
        permissions: [randomIntBetween(1, 3), randomIntBetween(4, 6), randomIntBetween(7, 10)],
        operation: randomIntBetween(1, 10000)
    }

    // trigger the test by groups sequentialy
    group('getAll', () => {
        test(params, scenario)
    })
}
```

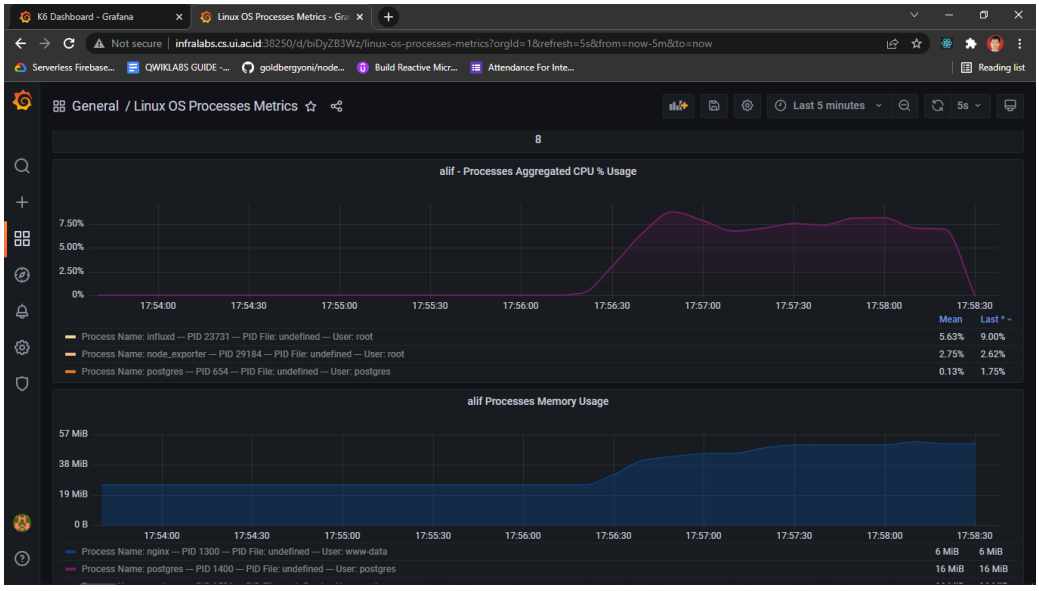
5. Memindahkan folder performance test ke GCP
6. Menjalankan performance test

Hasil Pengujian dan Analisis

Load Testing

Hasil





Avg Response Time = 1.38s
Peak Response Time = 10.4s
Error rates = 0%
CPU Utilization = Stabil 7%
Memory Utilization = 38 - 54 MiB

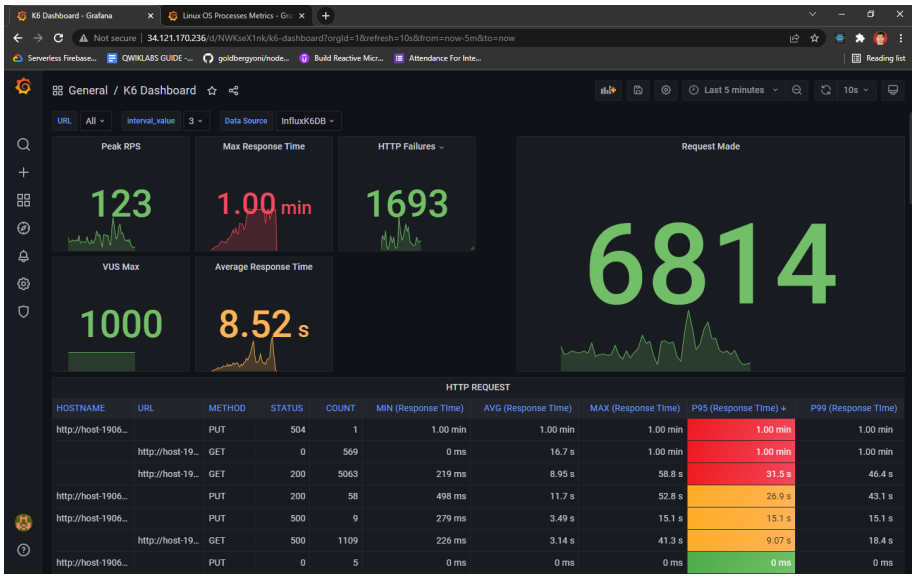
Analisis

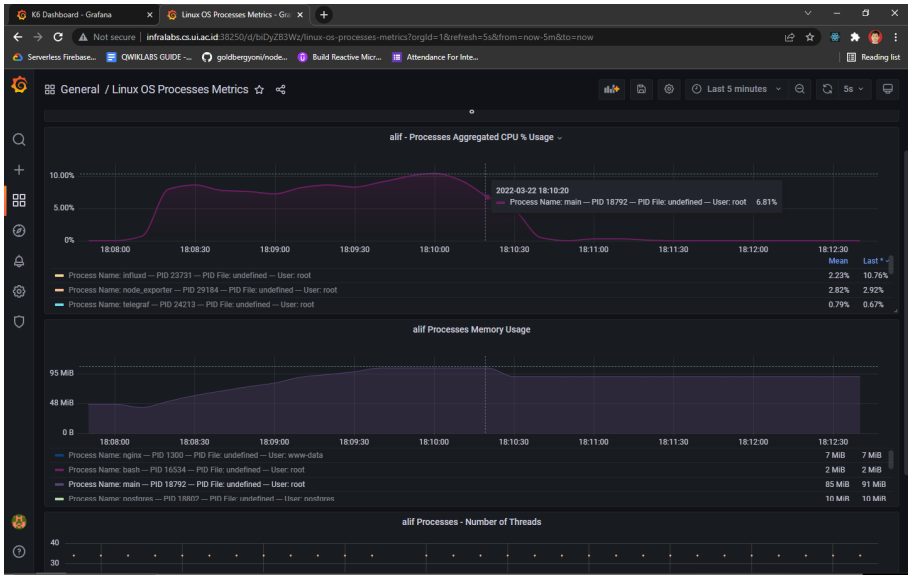
Load testing dilakukan selama 1 menit dengan 100 virtual users (VUs) yang meningkat secara incremental. Peak RPS 86.7 tercapai di awal-awal, yaitu ketika VUs masih belasan. Ketika VUs sudah mencapai angka 40 ke atas, performa turun menjadi stabil di +- 40 RPS.

Maksimum response time dicapai oleh endpoint PUT dengan waktu 10.4 s. Hal ini terjadi ketika VUs di angka 100 dan server dibanjiri oleh request. Untuk penggunaan memori dan cpu sendiri ternyata tidak terlalu memakan resource, yaitu hanya 7% penggunaan CPU dan 54 MiB penggunaan memori.

Stress Testing

Hasil





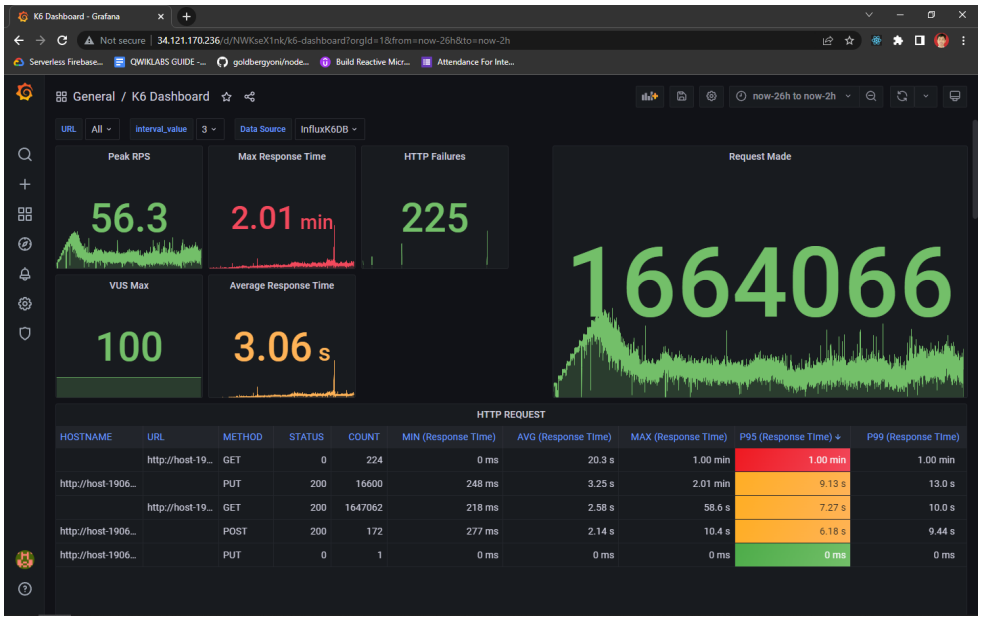
Avg Response Time = 8.52s
Peak Response Time = 60s
Error rates = 24%
CPU Utilization = 8% - 10%
Memory Utilization = 48 - 96 MiB

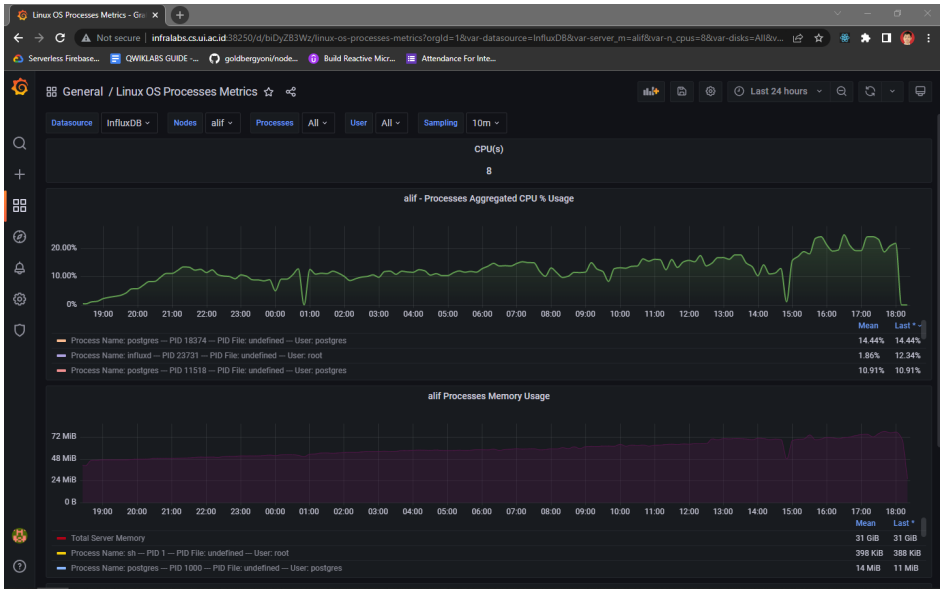
Analisis

Stress test dijalankan skenario 1000 VUs dalam waktu 1 menit. Sistem memberikan error 500 ketika VUs mencapai angka 800. Ketika VUs mendekati angka 1000, response yang dikembalikan adalah 504 (Gateway Time Out), yang kemungkinan dihasilkan oleh Nginx selaku reverse proxy. Sehingga dapat disimpulkan beban maksimum dari webservice ini adalah ~800 users yang mengakses bersamaan.

Endurance Testing

Hasil





Avg Response Time = 3.04s
Peak Response Time = 121s
Error rates = 0.013%
CPU Utilization = Stabil 15%, peak 24.80%
Memory Utilization = 48 - 75 MiB

Analisis

Endurance testing dilakukan dengan skenario 100 VUs selama 24 jam. Sebagaimana yang telah diperkirakan pada load testing, puncak RPS terjadi pada pukul 21.50 (56 RPS), yaitu ketika VUs mencapai angka 15. Setelah lebih dari 20, response time meningkat dan RPS stabil di angka 20 RPS.

Utilisasi resource dinilai cukup optimal, dengan peak cpu usage 24.80% dan memory utilization 75 MiB. Response time meningkat seiring berjalannya waktu, karena terdapat 172 request POST dikirim, artinya terdapat total penambahan 172 data.

Peak response time mencapai waktu 2 menit, yaitu pada endpoint PUT. Belum diketahui penyebab pasti dari insiden ini. Endurance test berjalan selama 24 jam dan mampu melayani hingga 1.664.066 request, hanya dengan kegagalan sebanyak 225 request.

Melalui endurance test ini, disimpulkan bahwa layanan webservice yang dibuat sebenarnya sudah mampu melayani load normal (+- 100 VUs) yang mengakses secara bersamaan dan terus menerus. Meski terkadang terjadi lonjakan response time, average response time 3s masih cukup masuk akal untuk load sebanyak itu.

Rencana Improvement

Salah satu solusi yang terpikir adalah penggunaan cache (seperti misalnya Redis) untuk endpoint GET. Semisal data yang ingin didapatkan ada di cache, maka akan diambil dari cache. Setiap ada perubahan data (PUT atau POST), maka cache juga akan diperbaharui. Memang akan menambah sedikit latency untuk aktivitas PUT dan POST, namun akan memberikan performa yang lebih baik untuk endpoint GET yang jauh lebih sering diakses.