Perbandingan Performa Kinerja Node.js, PHP, dan Python dalam Aplikasi REST

Performance Comparison of Node.js, PHP, and Python Performance

[1]Anugerah Christian Rompis, [2]Rizal Fathoni Aji
[1]Progam Studi Magister Teknologi Informasi, Universitas AMIKOM, Yogyakarta
[2]Fakultas Ilmu Komputer, Universitas Indonesia, Depok
e-mail: [1]starleaf1@gmail.com

Abstrak

Dengan berkembangnya jumlah aplikasi yang terhubung ke Internet, baik desktop maupun mobile, kebutuhan akan aplikasi server yang mampu menyediakan performa yang baik semakin meningkat. Salah satu cara mengoptimalkan performa server adalah dengan memakai bahasa pemrograman yang tepat. Dalam penelitian ini, tiga bahasa pemrograman yang terkait erat dengan web yakni Node.js, PHP, dan Python dibandingkan performanya dengan cara mengukur kecepatan respon serta konsumsi sumber daya komputasi. Dari penelitian ini didapati bahwa Node.js memiliki durasi respon yang paling cepat, sedangkan PHP adalah bahasa pemrograman yang memiliki performa umum yang paling baik.

Keywords : Kinerja, Node.Js, Python, PHP, Aplikasi REST

Abstract

With the increasing amount of Internet-connected applications, both on desktop and mobile devices, the need for high-performance server application also increases. One way to improve the performance of the app is by using the best programming language for the server. In this research, three programming languages commonly associated with web, Node.js, PHP, and Python is compared by measuring their response duration and resource consumption. This research revealed that Node.js has the fastest raw response speed, while PHP has the best all-round performance.

Keywords : Performance, Node.Js, Python, PHP, REST Application

1. PENDAHULUAN

Dalam rangka menangani peningkatan jumlah pengguna yang terhubung ke Internet, teknologi server web terus dikembangkan. Salah satu teknologi server web yang dimaksud adalah bahasa pemrograman server. Tidak hanya untuk meningkatkan kemudahan pembuatan aplikasi, tetapi juga untuk meningkatkan performa dari aplikasi tersebut.

Kecepatan aplikasi server menjadi semakin penting. Aplikasi yang kinerjanya baik adalah aplikasi yang dapat menangani banyak *request* dari *clients* tanpa mengonsumsi banyak sumber daya komputasi dari perangkat server, dan dengan demikian menekan biaya yang harus dikeluarkan oleh pengelola aplikasi tersebut. Saat ini terdapat beberapa bahasa pemrograman

yang populer dalam pengembangan aplikasi web. Ketiga bahasa tersebut yakni PHP, Node.js, dan Python.

Penelitian ini bertujuan mengukur serta membandingkan performa antara Python, Node.js, serta PHP. Diharapkan penelitian ini dapat membantu para pengambil keputusan dalam menentukan bahasa pemrograman yang tepat dalam rangka membangun sebuah aplikasi web, khususnya aplikasi yang berbasis *Representational State Transfer* (REST).

Aspek performa yang diukur yakni penggunaan CPU, penggunaan RAM, serta kecepatan respon. Keamanan, kemudahan pengembangan, serta skalabilitas aplikasi berada di luar batasan penelitian ini. Penelitian ini juga berfokus pada performa aplikasi *server*. Performa aplikasi *client* berada di luar batasan penelitian ini.

Terdapat beberapa penelitian yang serupa. Ueda dan Ohara [1] melakukan pembandingan antara bahasa pemrograman yang dikompilasi secara statis seperti Java, terhadap bahasa-bahasa yang dikompilasi secara dinamis seperti [2] menyelidiki kelayakan JavaScript sebagai bahasa pemrograman untuk dipakai menjadi sebuah server web. [3] melakukan penelitian yang membandingkan berbagai framework PHP serta PHP tanpa framework. [4] mengadakan penelitian yaitu membandingkan PHP, Python, dan Ruby. [5] mengadakan penelitian yang membandingkan kemampuan PHP dengan C dan Java untuk digunakan dalam *web service* berbasis SOAP.

2. METODE PENELITIAN

2.1 Lingkungan Percobaan

Penilitian ini dilaksanakan dengan dua unit komputer. Semua aplikasi obyek penelitian dipasangkan pada satu komputer yang bertindak sebagai server. Sementara, komputer kedua akan berperan sebagai client dan bertugas membangun serta mengirimkan request kepada komputer server. Hal ini ditujukan agar proses pembangunan request tidak mempengaruhi kinerja dari aplikasi-aplikasi yang akan diuji. Komputer server dan client akan dihubungkan melalui jaringan Ethernet LAN yang tidak terhubung ke host manapun kecuali kedua komputer tersebut di atas.

Komputer yang akan dijadikan server memiliki CPU Intel Core i3-5005U quad-core 2GHz, RAM 12GB, serta sistem operasi Ubuntu 16.04 LTS. Sementara komputer client memiliki CPU Intel Core i5, RAM 4GB, serta sistem operasi Ubuntu 16.04 LTS.

2.2 Aplikasi Obyek Penelitian

Aplikasi yang akan dijadikan aplikasi obyek penelitian adalah sebuah aplikasi informasi jasa transportasi. Pengguna aplikasi diwajibkan melakukan login sebelum dapat mengakses sebagian dari informasi yang disediakan oleh aplikasi tersebut. Otentikasi pengguna dilakukan dengan metode username dan password. Password disimpan dalam basis data dalam bentuk hash. Algoritma hash yang dipakai adalah Bcrypt versi 1. Pengguna yang sudah login dengan username dan password akan diberikan sebuah token untuk dipakai mengakses informasi-informasi terbatas yang disebut di atas. Token ini disusun dengan menggunakan JSON Web Token (JWT). JWT yang sudah diberikan harus disertakan dalam *Authorization request header* yang dikirimkan oleh *client* agar dapat memperoleh informasi yang mewajibkan otentikasi.

Aplikasi ini menyediakan informasi berdasarkan URL yang diminta oleh *client*. Informasi tersebut akan disajikan dalam format *JavaScript Object Notation* (JSON). Tabel 1 menunjukkan URL yang dilayani oleh aplikasi, informasi yang didapat melalui URL tersebut, beserta keterangan-keterangan lain.

Tabel 1 URL dalam aplikasi

URL	Tujuan	Keterangan
/api/	Mengetahui status aplikasi	Balasan yang diterima dari server adalah "*OK*" apabila aplikasi berjalan dengan baik.
/api/login/	Melakukan login	*Client* harus mengirimkan sebuah *POST request* yang di dalamnya berisi *fields* yakni *username* dan *password*. Apabila login sukses, maka aplikasi akan mengirimkan JWT sebagai balasan.
/api/schedule/	Mendapatkan jadwal keberangkatan kendaraan	
/api/vehicle/	Mendapatkan daftar kendaraan yang tersedia	Wajib login terlebih dahulu.
/api/vehicle/{id}	Mendapatkan informasi mendetil mengenai kendaraan tertentu	*{id}* adalah kode identifikasi kendaraan dalam sistem berupa *string* heksadesimal. *{id}* diperoleh melalui */api/vehicle/*. Wajib login terlebih dahulu.
/api/driver/	Mendapatkan daftar pengemudi yang tersedia	Wajib login terlebih dahulu.
/api/driver/{id}	Mendapatkan informasi mendetil mengenai pengemudi tertentu.	*{id}* adalah kode identifikasi pengemudi dalam sistem berupa *string* heksadesimal. *{id}* diperoleh melalui */api/driver/*. Wajib login terlebih dahulu.

Perangkat sistem pengelolaan basis data yang akan digunakan untuk ketiga aplikasi adalah MySQL versi 14.14.

Aplikasi yang dibangun untuk PHP akan dibangun di atas PHP versi 7.0.28. Aplikasi tersebut akan dibangun dengan menggunakan framework Slim versi 3. JWT dalam aplikasi ini diimplementasi dengan library Firebase PHP-JWT versi 5.0.0. Akses basis data dilayani oleh PHP Data Objects (PDO) yang tersedia pada PHP versi ini.

Aplikasi Python akan dibangun di atas Python versi 2.7.12. Aplikasi tersebut akan dibangun dengan framework Flask versi 0.12.2. Koneksi dengan basis data disediakan oleh library PyMySQL versi 0.8.0. Sementara implementasi JWT dan Bcrypt dilakukan melalui modul jwt versi 1.6.1 dan Bcrypt versi 3.1.4.

Aplikasi Node.js akan dibangun pada Node versi 6.14.1 dengan menggunakan framework Express versi 4. Algoritma Bcrypt dalam aplikasi ini diimplementasi dengan modul bcrypt-nodejs versi 0.0.3. Otentikasi JWT ditangani oleh modul passport-jwt versi 4.0.0.

2.3. Perangkat Pengambilan Data

Pengukuran kecepatan respon akan dilakukan dengan menggunakan JMeter. JMeter adalah perangkat lunak yang dapat menghasilkan dan mengirimkan requests pada sebuah server. JMeter dapat dikonfigurasikan untuk mengirimkan berbagai tipe, volume, pengaturan waktu, serta isi request. JMeter juga dapat menyajikan hasil pengujian berupa tabel dan grafik. Dalam penelitian ini, JMeter akan dikonfigurasi untuk mengirimkan 10000 request pada saat yang bersamaan.

Pengukuran konsumsi CPU dan memori akan dilakukan dengan menggunakan program ps yang terdapat dalam Linux. Program tersebut akan dieksekusi setiap 0,5 detik dan outputnya akan disaring dengan program grep sehingga hanya akan menampilkan proses yang relevan serta disalurkan ke dalam sebuah berkas teks untuk kemudian dianalisa.

2.4 Prosedur Pengambilan Data

Pengambilan data dilaksanakan dengan prosedur untuk tiap-tiap URL dalam tiap-tiap aplikasi obyek penelitian sebagai berikut:

1. Aplikasi dijalankan hingga siap menerima request dari client.
2. Program ps dijalankan pada komputer server.
3. Aplikasi JMeter dijalankan pada komputer client dan skenario pengujian dieksekusi.
4. Aplikasi server dihentikan dan komputer server dimatikan dan dihidupkan kembali untuk URL selanjutnya.

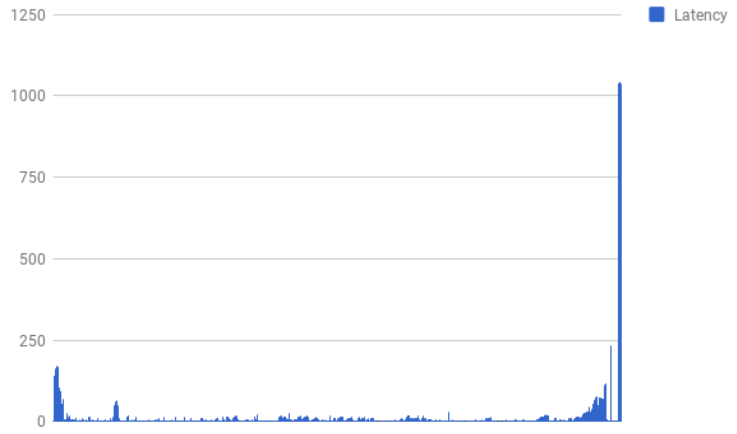
3. HASIL DAN PEMBAHASAN

3.1 Waktu Respon

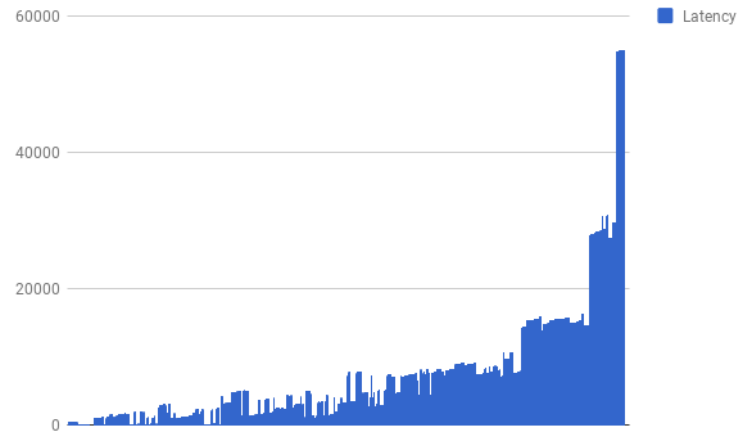
3.1.1 Status Aplikasi

Query terhadap status aplikasi melalui URL */api/* adalah tugas sederhana yang dilakukan tanpa perlu mengakses basis data maupun melaksanakan algoritma kriptografi yang rumit. *Request* terhadap URL ini diselesaikan oleh Node.js dengan rata-rata durasi 13,11 milidetik (Gambar 1). PHP memberikan respon dalam rata-rata 663,016 milidetik, namun gagal memberi respon terhadap 96 *requests* (Gambar 2). Python memberikan respon dalam rata-rata 16012,724 milidetik, namun gagal memberi respon terhadap 400 *requests* (Gambar 3).

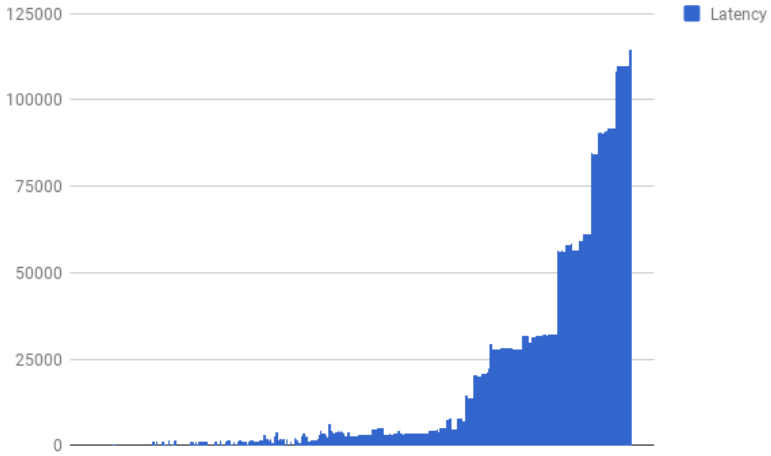
Dalam rangka merespon *request* mengenai status aplikasi, Node.js memakan rata-rata 26,14% CPU dan 0,56% memori. PHP memakai rata-rata 33,79% CPU dan 0,2% memori. Python mengonsumsi rata-rata 9,68% CPU dan 0,15% memori (Gambar 4).



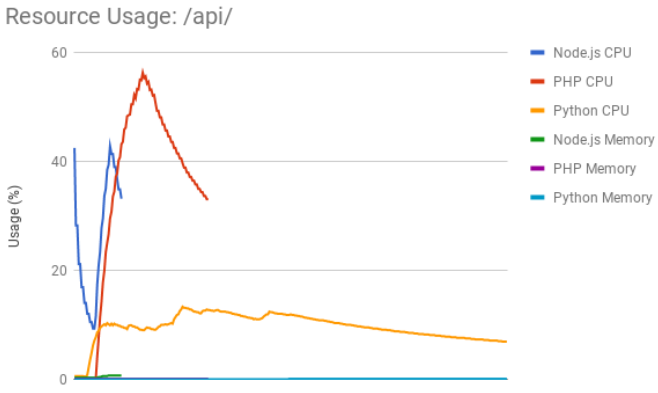
Gambar 1 Waktu respon Node.js untuk URL */api/*



Gambar 2 Waktu respon PHP untuk URL */api/*



Gambar 3 Waktu Respon Python untuk URL */api/*



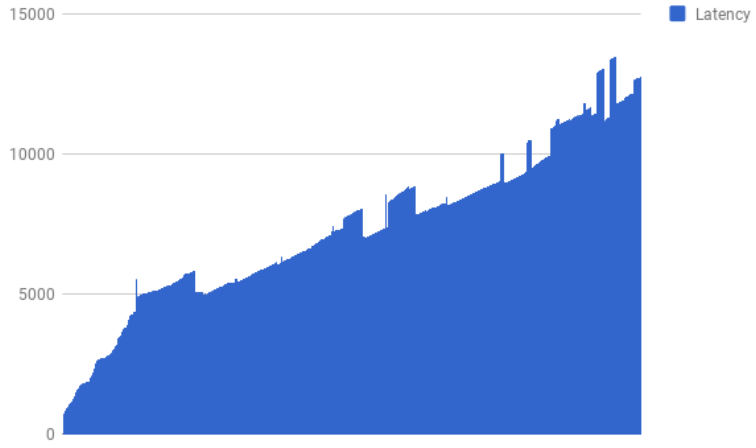
Gambar 4 Konsumsi Sumber Daya untuk URL */api/*

3.1.2 Daftar Pengemudi

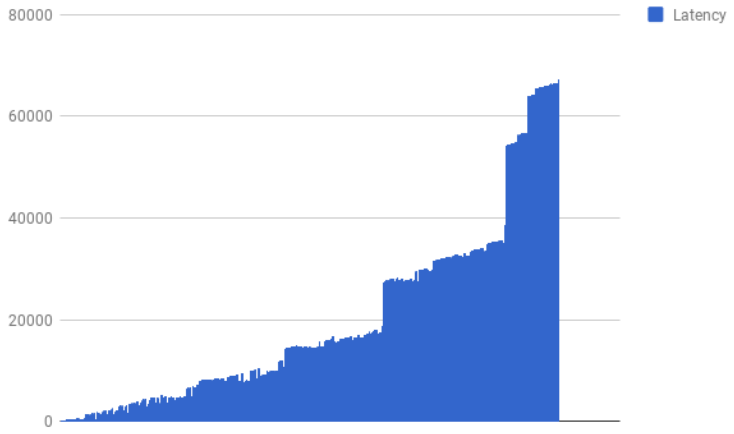
Query daftar pengemudi adalah salah satu fitur aplikasi yang hanya bisa diakses dengan otentikasi. Node.js memberikan respon terhadap *requests* dari *client* rata-rata dalam 7278,24 milidetik (Gambar 5). PHP memberikan respon rata-rata 84 milidetik, namun gagal memberi

respon terhadap 1091 *request* (Gambar 6). Python memberikan respon dalam rata-rata 42254,89 milidetik namun gagal memberi respon terhadap 3303 *requests* (Gambar 7).

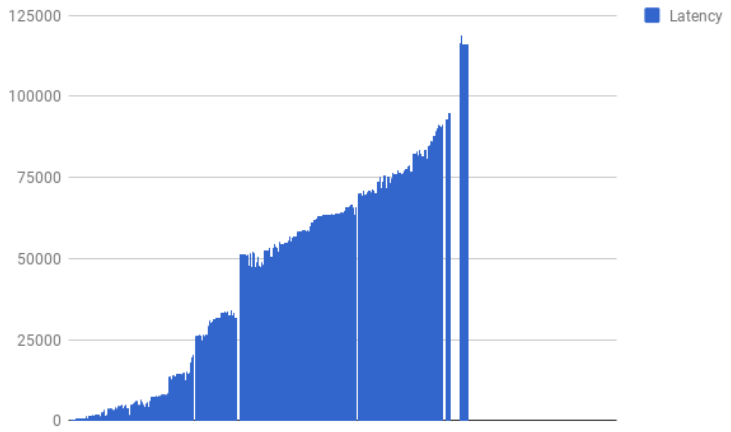
Dalam rangka merespon *request* mengenai daftar pengemudi, Node.js mengonsumsi rata-rata 47,67% CPU dan 1,82 memori. PHP mengonsumsi rata-rata 35,85% CPU dan 0,2% memori. Python mengonsumsi rata-rata 16,26% CPU dan 0,2% memori (Gambar 8).



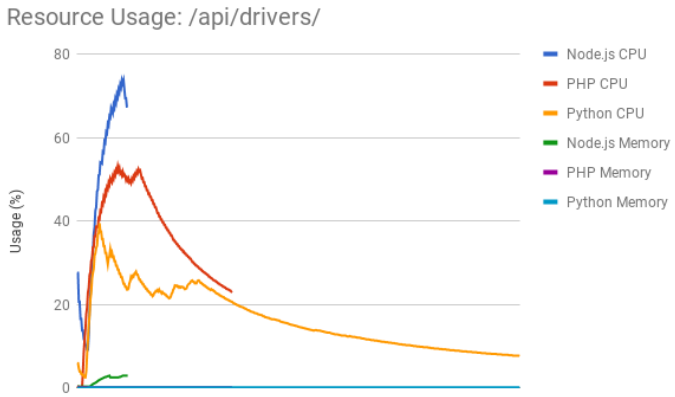
Gambar 5 Waktu Respon Node.js untuk URL */api/driver/*



Gambar 6 Waktu respon PHP untuk URL */api/driver/*



Gambar 7 Waktu respon Python untuk URL */api/driver/*

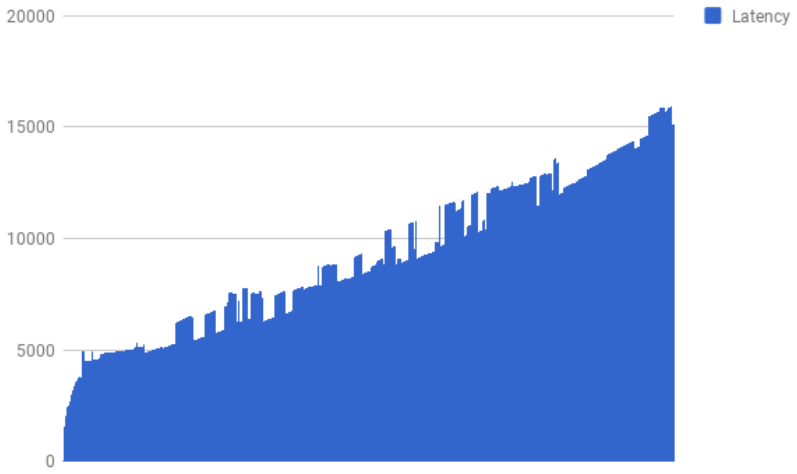


Gambar 8 Penggunaan Sumber Daya untuk URL /api/driver/

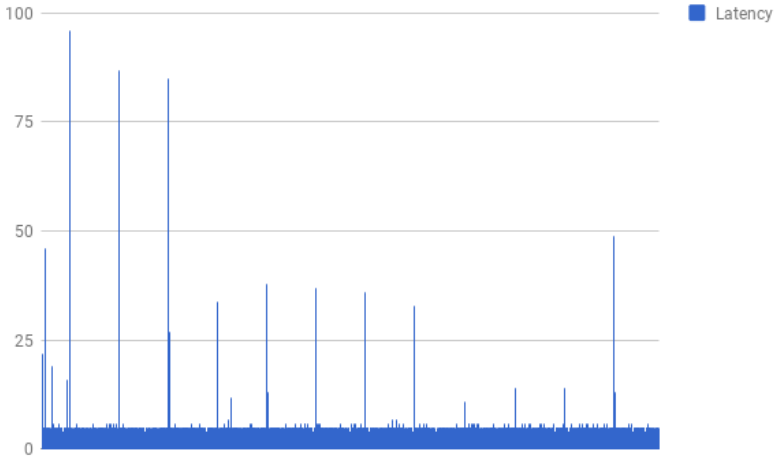
3.1.3 Detail Pengemudi

Query terhadap detail pengemudi melibatkan verifikasi JWT dan pengambilan 1 baris dari basis data. Node.js memberikan respon rata-rata dalam 9157,99 milidetik (Gambar 9). PHP memberikan respon rata-rata dalam waktu 3,95 milidetik (Gambar 10). Python memberikan respon rata-rata dalam waktu 4,31 milidetik (Gambar 11).

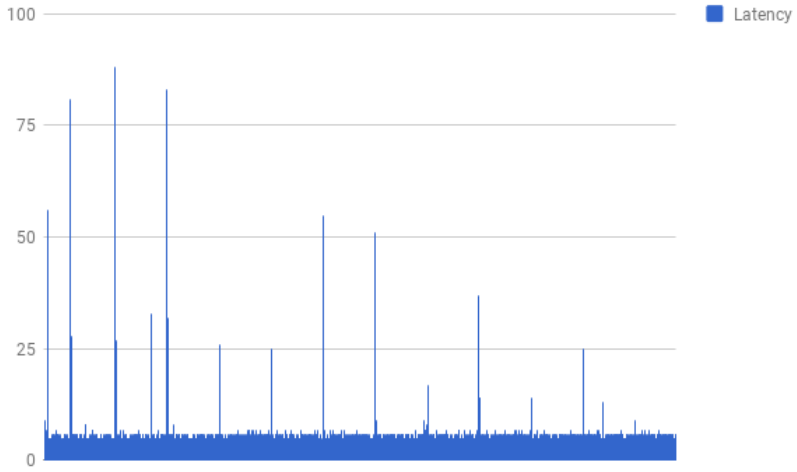
Untuk URL */api/driver/{id}* Node.js mengonsumsi rata-rata 45,37% CPU dan 1,81% memori. PHP mengonsumsi rata-rata 29,78% CPU dan 0,2% memori. Sementara Python mengonsumsi 8,97% CPU dan 0,1% memori (Gambar 12).



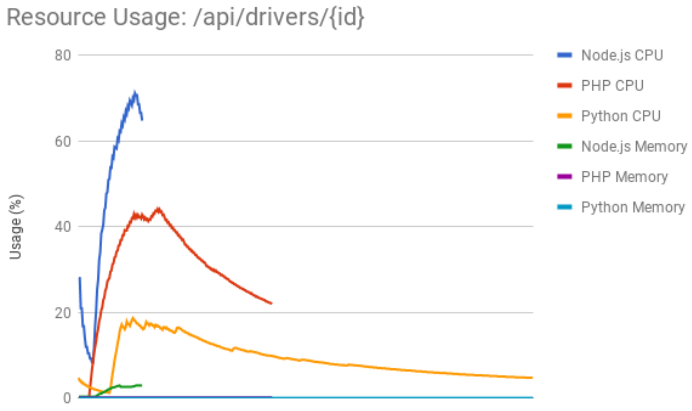
Gambar 9 Waktu respon Node.js untuk URL */api/driver/{id}*



Gambar 10 Waktu respon PHP untuk URL */api/driver/{id}*



Gambar 11 Waktu respon Python untuk URL */api/driver/{id}*

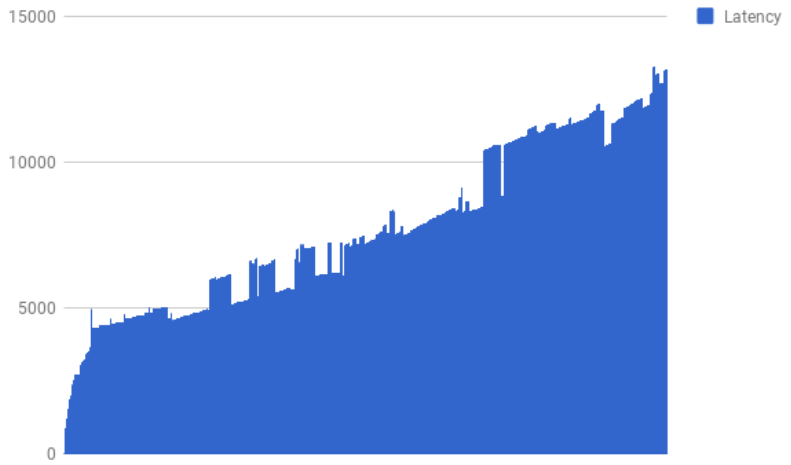


Gambar 12 Konsumsi Sumber Daya untuk URL */api/driver/{id}*

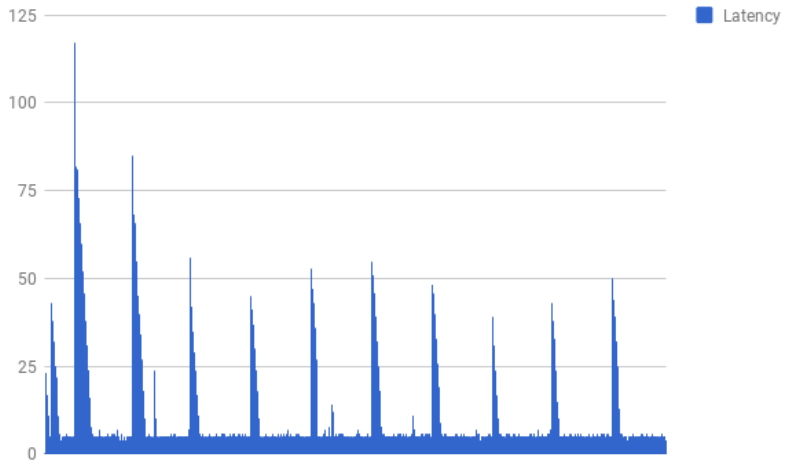
3.1.4 Daftar Kendaraan

Daftar kendaraan, seperti halnya daftar pengemudi, adalah sebuah tugas yang mewajibkan *client* menyertakan JWT dalam header dan melibatkan pengambilan beberapa baris dari basis data. Node.js memberikan respon dalam waktu rata-rata 7632,92 milidetik (Gambar 13). Rata-rata waktu respon PHP adalah 7,8 milidetik (Gambar 14). Sementara rata-rata waktu respon Python adalah 10 milidetik (Gambar 15).

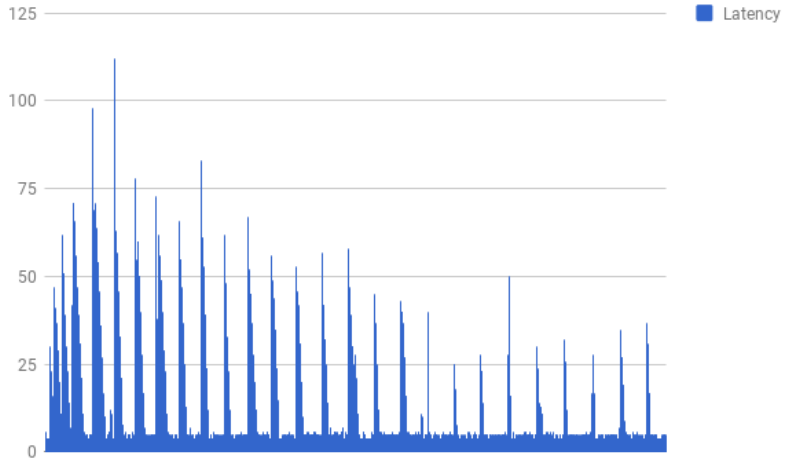
Dari segi konsumsi sumber daya, Node.js mengonsumsi rata-rata 31,57% CPU dan 1,49% memori. PHP mengonsumsi rata-rata 33,3% CPU dan 0,2% memori. Python mengonsumsi rata-rata 13,32% CPU dan 0,1% memori (Gambar 16).



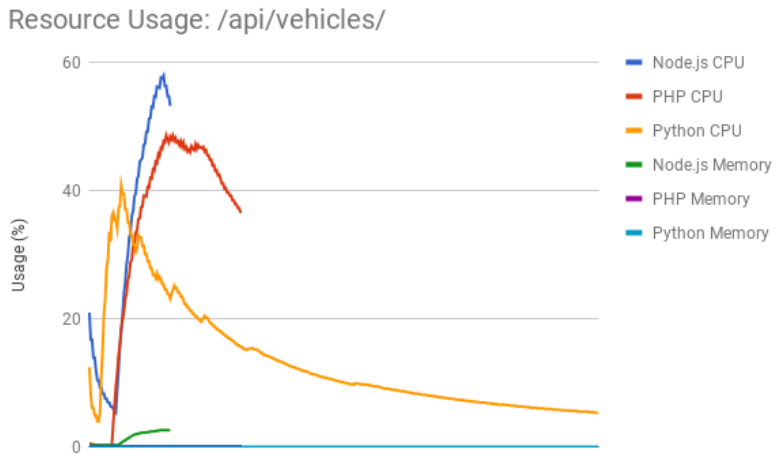
Gambar 13 Waktu respon Node.js untuk URL */api/vehicle/*



Gambar 14 Waktu respon PHP untuk URL */api/vehicle/*



Gambar 15 Waktu respon Python untuk URL */api/vehicle/*

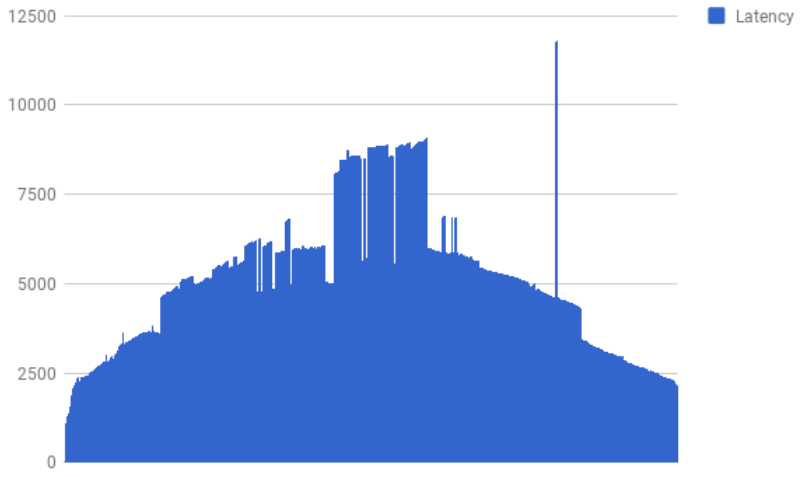


Gambar 16 Konsumsi Sumber Daya untuk URL */api/vehicle*

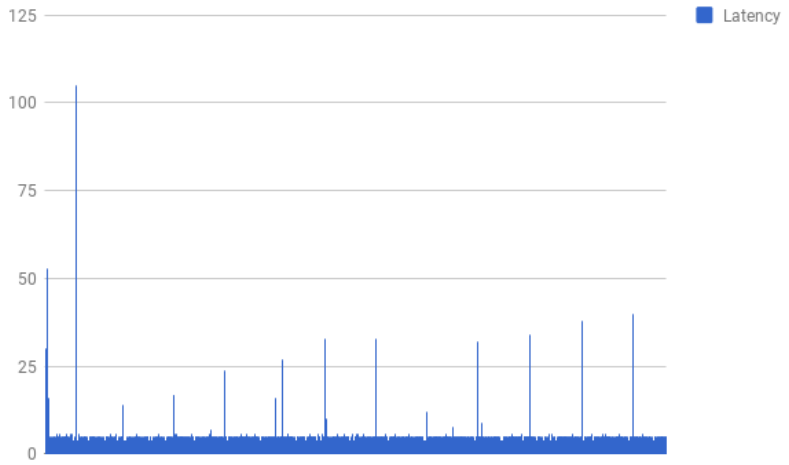
3.1.5 Detail Kendaraan

Query terhadap detail sebuah kendaraan tertentu melibatkan verifikasi JWT. Node.js memberi respon dalam waktu rata-rata 4946,07 milidetik (Gambar 17). PHP memberi respon rata-rata dalam 3,77 milidetik (Gambar 18). Python memberi respon dalam waktu rata-rata 3,53 milidetik (Gambar 19).

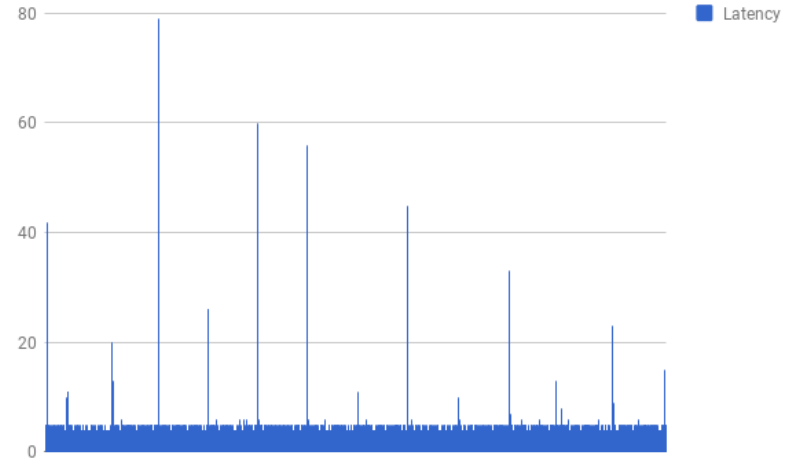
Untuk menyajikan informasi mendetail mengenai kendaraan yang spesifik, Node.js mengonsumsi rata-rata 51,76% CPU dan 1,9% memori . PHP mengonsumsi rata-rata 33,72% CPU dan 0,2% memori. Python mengonsumsi rata-rata 13,67% CPU dan 0,19% memori (Gambar 20).



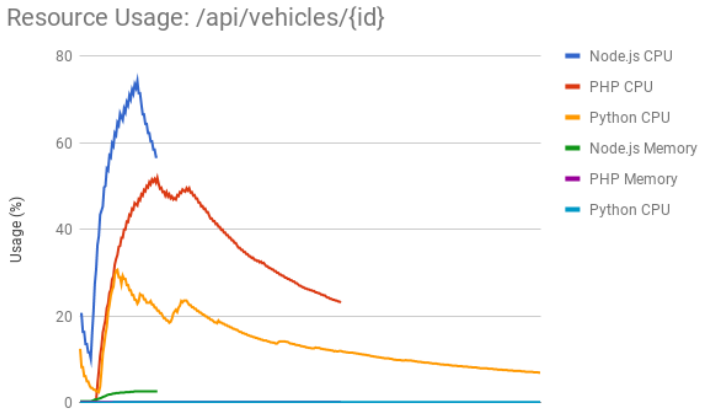
Gambar 17 Waktu respon Node.js untuk URL */api/vehicle/{id}*



Gambar 18 Waktu respon PHP untuk URL */api/vehicle/{id}*



Gambar 19 Waktu respon Python untuk URL */api/vehicle/{id}*

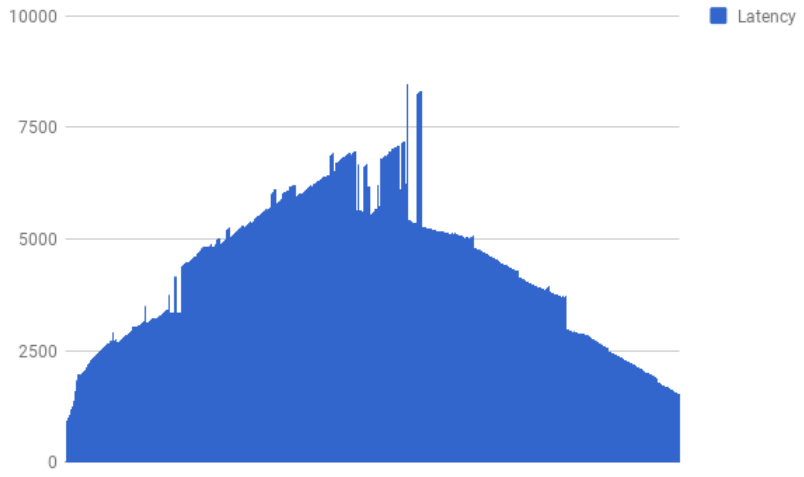


Gambar 20 Konsumsi Sumber Daya untuk URL */api/vehicles/{id}*

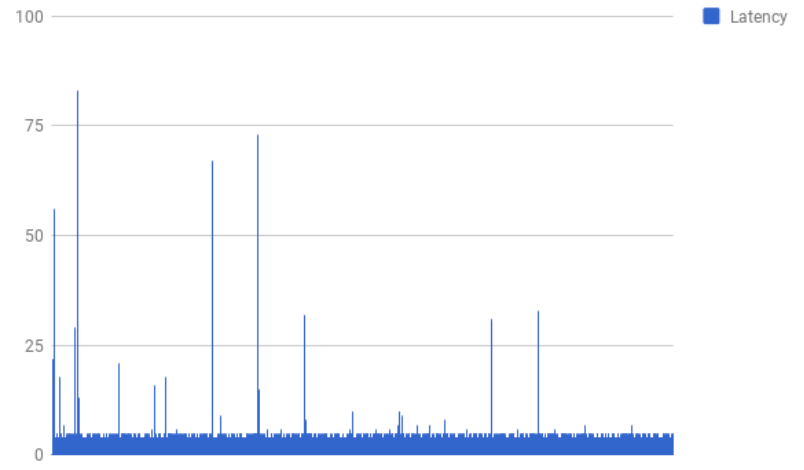
3.1.6 Jadwal Keberangkatan

Jadwal keberangkatan adalah salah satu URL yang tidak mewajibkan pengguna untuk login terlebih dahulu, dan dengan demikian tidak melibatkan perhitungan kriptografi, baik untuk memverifikasi JWT maupun membandingkan *hash*. Tugas ini melibatkan akses terhadap basis data. Waktu respon Node.js untuk *query* ini rata-rata adalah 4207,28 milidetik (Gambar 21). PHP memberikan respon dalam waktu 3,64 milidetik (Gambar 22). Python memberikan respon dalam waktu rata-rata 4,17 milidetik (Gambar 23).

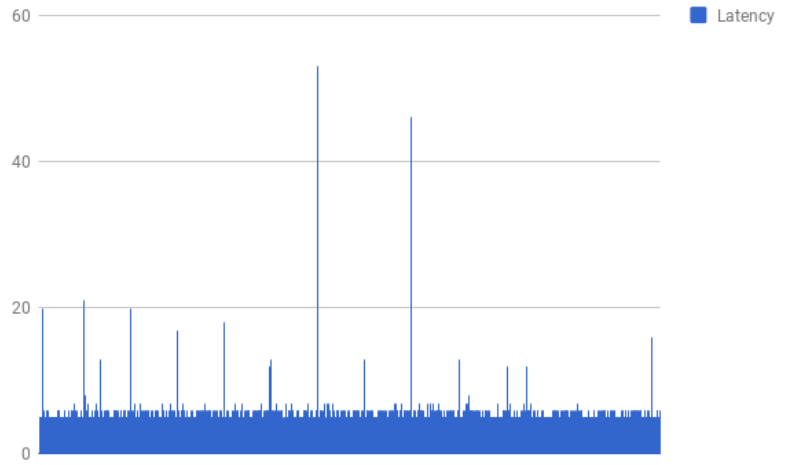
Untuk menampilkan informasi jadwal keberangkatan kendaraan Node.js mengonsumsi rata-rata 42,52% CPU dan 1,72% memori. PHP mengonsumsi rata-rata 42,2% CPU dan 0,2% memori. Python mengonsumsi rata-rata 13,87% CPU dan 0,2% memori (Gambar 24).



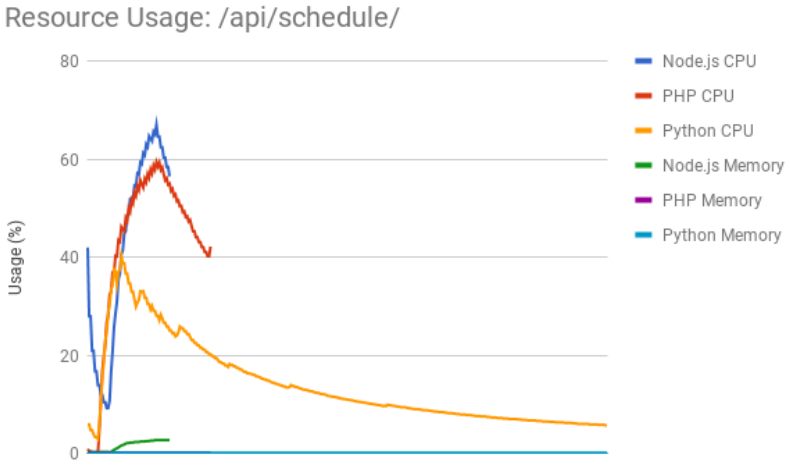
Gambar 21 Waktu respon Node.js untuk URL */api/schedule/*



Gambar 22 Waktu respon PHP untuk URL */api/schedule/*



Gambar 23 Waktu respon Python untuk URL */api/schedule/*

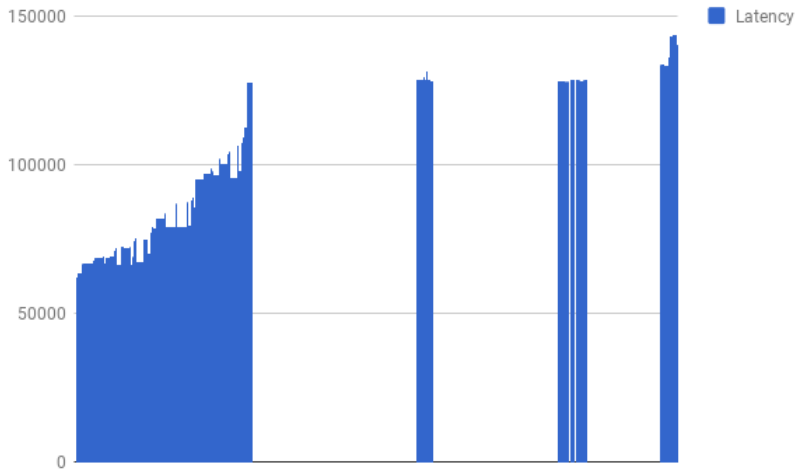


Gambar 24 Konsumsi Sumber Daya untuk URL */api/schedule/*

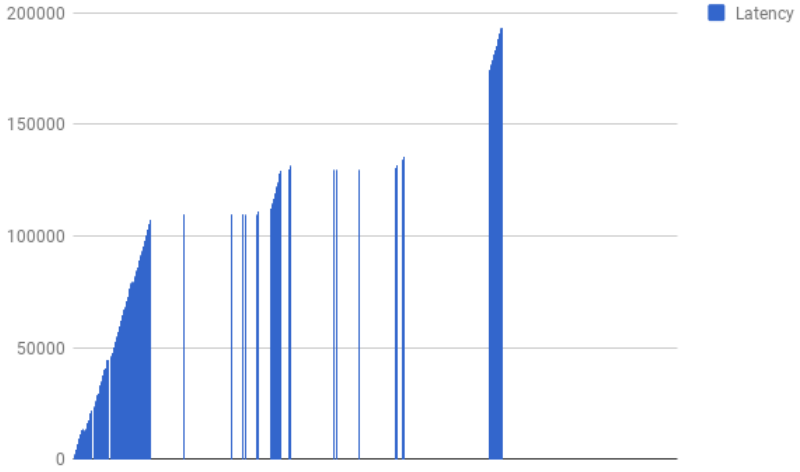
3.1.7 Login

Proses login melibatkan akses basis data sekaligus eksekusi algoritma Bcrypt untuk membandingkan *hash* password. Bcrypt adalah sebuah algoritma kriptografi yang relatif lebih rumit dibandingkan verifikasi JWT. Node.js memberikan respon dalam waktu rata-rata 91514,27 milidetik dan gagal memberi respon terhadap 9663 *request* (Gambar 25). PHP memberi respon dalam waktu rata-rata 80892,66 milidetik dan gagal memberi respon terhadap 8480 *request* (Gambar 26). Python memberi respon dalam waktu rata-rata 50334,36 milidetik dan gagal memberi respon terhadap 9125 *request* (Gambar 27).

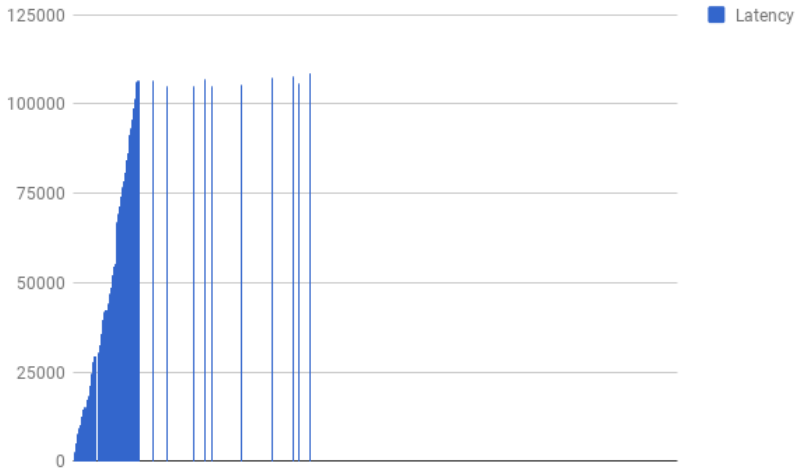
Untuk melakukan login, Node.js mengonsumsi sumber daya rata-rata sebesar 79,89% CPU dan 0,84% memori. PHP mengonsumsi sumber daya rata-rata sebesar 42,2% CPU dan 0,2% memori. Python mengonsumsi 61,07% CPU dan 0,19% memori (Gambar 28).



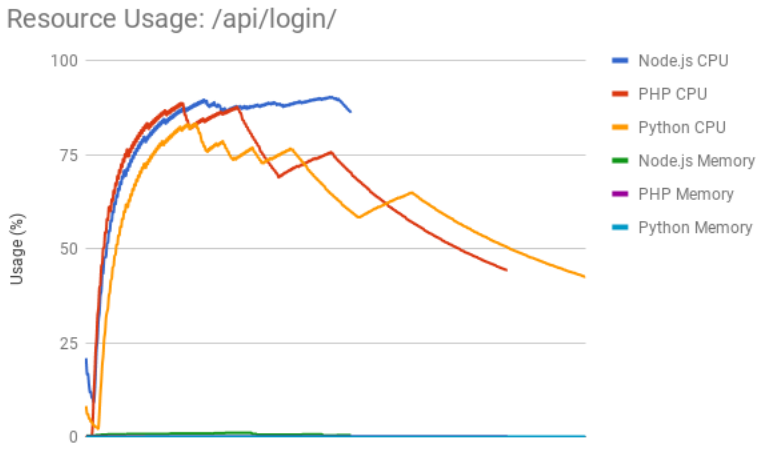
Gambar 25 Waktu respon Node.js untuk URL */api/login/*



Gambar 26 Waktu respon PHP untuk URL */api/login/*



Gambar 27 Waktu respon Python untuk URL */api/login/*



Gambar 28 Konsumsi Sumber Daya untuk */api/login/*

4. KESIMPULAN

Berdasarkan data-data yang didapat dari percobaan yang telah dipaparkan di atas, maka dapat ditarik kesimpulan-kesimpulan sebagai berikut:

1. Dari ketiga obyek penelitian, Node.js adalah bahasa pemrograman yang paling cocok untuk melaksanakan tugas-tugas yang tidak melibatkan perhitungan-perhitungan kriptografi yang intensif. Khususnya menyediakan respon langsung terhadap *request* yang masuk.
2. PHP menyediakan performa yang paling konsisten di antara ketiga aplikasi, serta menyediakan keseimbangan antara konsumsi sumber daya dan kecepatan respon.

5. SARAN

Penelitian selanjutnya dapat dilaksanakan dengan melibatkan berbagai *framework* yang berbeda, modul-modul Bcrypt atau JWT yang berbeda, maupun penghubung basis data yang berbeda.

UCAPAN TERIMAKASIH

Penulis mengucapkan terima kasih kepada Universitas AMIKOM dan PJJ APTIKOM yang telah memungkinkan pelaksanaan penelitian ini.

DAFTAR PUSTAKA

[1] Y. Ueda, M. Ohara, Performance competitiveness of a statically compiled language for server-side Web applications. *International Symposium on Performance Analysis of Systems and Software (ISPASS*), California, April 24-25, 2017.

[2] I. K. Chaniotis, K. I. D. Kyriakou, dan N. D Tselikas, Is Node. js a viable option for building modern web applications? A performance evaluation study, *Computing*, No.10, Vol.97, 1023-1044, 2015.

[3] R. Das, dam L. P. Saika, L, Comparison of Procedural PHP with Codeigniter and Laravel Framework. *International Journal of Current Trends in Engineering & Research*, no.6, vol.2, 2016.
[4] K. Purer, PHP vs. Python vs. Ruby–The web scripting language shootout, Vienna University of Technology, Vienna, 2009.
[5] T. Suzumura, S. Trent, M. Tatsubori, A. Tozawa, T. Onedera. Performance comparison of web service engines in php, java and c. *International Conference on Web Service (ICWS'08)*, Beijing, September 23-26, 2008.