Jurnal Riset Sistem Informasi Dan Teknik Informatika (JURASIK)
Volume 8, Nomor 1, Februari 2023, pp 155-168
Terakreditasi Nomor 204/E/KPT/2022 | ISSN: 2527-5771/EISSN: 2549-7839
https://tunasbangsa.ac.id/ejurnal/index.php/jurasik

Analisis Performa Backend Framework: Studi Komparasi Framework Golang dan Node.js

Suwarno[1], Amalia Putri Yulandi[2]
[1,2]Universitas Internasional Batam, Indonesia
Email: Suwarno.liang@uib.ac.id[1], 1931025.amalia@uib.edu[2]

Abstract

Currently, there are many programming languages that have been developed to assist developers in creating applications. One of them is the Golang programming language. Golang is a programming language created and developed by the Google Engineer team in 2009. The Golang programming language has several uses including as a language for building Backend Stacks, developing E-Commerce applications, and developing Cloud Native. In this case, the focus is on the Golang programming language as the back-end stack. To find out the effectiveness of the Golang programming language, a comparison is made with another programming language, namely the Node.js programming language. In this research, the two programming languages are compared with parameters of response speed, consumption of CPU and memory resources. This research reveals that Golang responds faster, and uses less CPU resources. Meanwhile, Node.Js has a good balance in the use of resources, especially memory.

Keywords*: Golang, Node.js, Backend, API, Comparison*

Abstrak

Bahasa pemrograman saat ini banyak sekali yang telah dikembangkan untuk membantu para pengembang dalam membuat sebuah aplikasi. Salah satunya adalah bahasa pemrograman Golang. Golang adalah bahasa pemrograman yang diciptakan dan dikembangkan oleh tim Engineer Google pada tahun 2009. Bahasa pemrograman Golang memiliki beberapa kegunaan diantaranya sebagai bahasa dalam membangun Backend Stack, pengembangan aplikasi E-Commerce, dan pengembangan Cloud Native. Dalam hal ini berfokus pada bagian bahasa pemrograman Golang sebagai back-end stack. Untuk mengetahui keefektifan pada bahasa pemrograman Golang dilakukan perbandingan dengan bahasa pemrograman lain yaitu bahasa pemrograman Node.js. Dalam penelitian ini, dua bahasa pemrograman tersebut dibandingkan dengan parameter kecepatan respon, konsumsi sumber daya CPU dan memori. Penelitian ini mengungkapkan bahwa Golang lebih cepat memberikan respon, dan penggunaan sumber daya CPU lebih sedikit. Sedangkan Node.Js memiliki keseimbangan yang baik dalam penggunaan sumber daya khususnya pada memori.

Kata kunci*: Golang, Node.js, Backend, API, Komparasi*

1. PENDAHULUAN

Pada saat ini, banyak sekali bahasa pemrograman yang dibuat untuk membantu para pengembang dalam membuat sebuah aplikasi. Salah satunya adalah bahasa pemrograman Golang. Golang adalah bahasa pemrograman yang diciptakan dan dikembangkan oleh tim *Engineer* Google pada tahun 2009. Awalnya bahasa tersebut hanya digunakan untuk kepentingan internal. Kemudian bahasa ini dirilis untuk kepentingan *public* dan bersifat *Open Source* sehingga siapapun bisa mengembangkan bahasa Go atau Golang. Bahasa pemrograman Golang memiliki beberapa kegunaan diantaranya sebagai bahasa dalam membangun *Backend Stack*, pengembangan aplikasi *E-Commerce*, dan pengembangan *Cloud Native*. Dalam hal ini kami berfokus pada bagian bahasa pemrograman Golang sebagai *Backend Stack*. *Backend* sendiri merupakan suatu bagian dari aplikasi yang

berjalan pada sisi server (*Server-Side*) bertugas menghubungkan antara sisi pengguna dengan basis data dalam manipulasi data ke tempat basis data, sehingga *Backend* tidak melakukan interaksi secara langsung kepada pengguna. Dalam meningkatkan interaksi antar layanan *client* yang berbeda dengan *server*, salah satunya memanfaatkan teknologi yang digunakan yaitu *Application Programming Interfaces* (API). API merupakan antarmuka yang digunakan untuk mengakses aplikasi atau layanan dari sebuah program. Sedangkan *Representational State Transfer API* (REST API) merupakan istilah yang dipakai untuk layanan web yang mengimplementasikan arsitektur REST sebagai API [1].

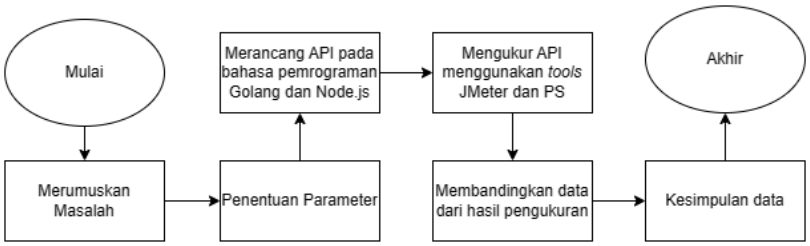
Golang memiliki beberapa kelebihan yang digunakan sebagai *Backend* yaitu mempunyai fitur *concurrency*, merupakan fitur yang dapat mengerjakan beberapa perintah dalam satu waktu secara bersamaan. Selain itu, Golang mempunyai fitur *Garbage Collector* sehingga tidak membutuhkan banyak memori saat menggunakannya [2]. Meskipun Golang terbilang masih baru, sudah banyak industri dan perusahaan menggunakan bahasa Golang hingga di tahap *Level Production*, Di Indonesia sendiri Golang sudah mulai banyak dikembangkan oleh kalangan *Developer* sebagai bahasa pemrograman *Backend Stack*. Diantaranya perusahaan terkemuka di Indonesia menggunakan Golang pada *Level Production* adalah Gojek dan Tokopedia. Untuk perusahaan terkemuka lainnya di luar Indonesia yang menggunakan Golang pada *Level Production* adalah Uber, Dropbox, termasuk Google sendiri. Dengan kepopulerannya penulis ingin melakukan sebuah penelitian mengenai bahasa pemrograman Golang sebagai *Backend Stack*.

Penulis menggunakan bahasa pemrograman Node.js sebagai perbandingan dengan bahasa Golang. Karena Node.js memiliki popularitas dilihat dari survei yang dilakukan oleh Stackflow pada tahun 2020, Node.js menduduki peringkat pertama dalam framework yang paling banyak digunakan [3]. Dan Node.js termasuk kedalam jajaran 10 bahasa pemrograman terpopuler di Indonesia menurut dari harian Kompas.com. Selain itu, Node.js memiliki kelebihan pada teknik *Non-Blocking* yang memungkinkan operasi-operasi dijalankan oleh sistem secara paralel tanpa harus menunggu operasi sebelumnya selesai sehingga memungkinkan banyak *request* dapat diselesaikan secara paralel [4]. Dari hal tersebut menjadikan penulis yakin melakukan penelitian untuk menemukan keefektifan menggunakan bahasa pemrograman golang dibanding Node.js sebagai *Backend Stack*. Hasil penelitian ini dapat memberikan pemahaman mengenai *Backend Stack* antara bahasa pemrograman Golang dan Node.js sehingga dapat menghasilkan keputusan yang berguna dalam membangun sebuah aplikasi yang tepat, khususnya aplikasi yang berbasis *Representational State Transfer* (REST).

2. METODOLOGI PENELITIAN

Penelitian ini menggunakan pendekatan metode eksperimen dengan diawali menentukan rumusan masalah yang diteliti lebih dalam. Kemudian penulis menentukan parameter yang digunakan sebagai pengukur dari penelitian yang dilakukan dalam komparasi antara bahasa pemrograman Golang dan Node.Js. Setelah itu, penulis membuat rancangan API pada bahasa pemrograman Golang dan Node.js. Setelah membuat rancangan API dari masing-masing bahasa

pemrograman, Penulis melakukan pengukuran dari API tersebut dengan menggunakan JMeter sebagai pengukuran kecepatan respon dan ps untuk mengukur penggunaan CPU dan RAM sehingga data tersebut digunakan dalam penelitian ini. Kemudian setelah mendapatkan data dari uji coba penulis melakukan analisis dan memperbandingkan hasil dari masing-masing bahasa pemrograman dan menentukan kelebihan apa saja yang ada pada bahasa pemrograman Golang dibanding dengan bahasa pemrograman Node.js. Kemudian penulis membuat sebuah kesimpulan dari tahap memperbandingkan data tersebut. Keseluruhan langkah-langkah ini terlihat pada Gambar 1.



Gambar 1. Alur Penelitian

Parameter yang digunakan untuk mengukur aspek performa dari setiap API masing-masing bahasa pemrograman Golang dan Node.Js sebagai penentu keefektifan apa saja yang dimiliki Golang dibanding Node.js dibagi menjadi 3 bagian yaitu, penggunaan CPU, penggunaan RAM, dan kecepatan respon [5].

Penelitian ini menggunakan metode *Scrum* dalam membangun aplikasi Review Film dengan membuat API pada tiap-tiap *framework* Golang dan Node.js. Pendekatan ini dipilih dikarenakan dapat membangun aplikasi dengan cara yang lebih cepat dan efisien. Menurut jurnal [6], di setiap tahap pengembangan terdapat aktivitas kerja yang terlingkup di dalam suatu pola proses yang disebut dengan *Sprint*. Pada pola ini pekerjaan akan lebih cepat dan efisien. Terbukti terdapat beberapa pola yang dapat mempercepat proses pembuatan aplikasi tersebut antara lain:

a) *Backlog*, pada tahapan ini melakukan pengumpulan fitur/API apa saja yang ditambahkan. Pada penelitian ini berikut API yang dibangun dalam obyek penelitian mengenai aplikasi "Review Film". Pengguna aplikasi diwajibkan melakukan *Login* sebelum mengakses. Sebagian informasi melalui aplikasi tersebut. Otentikasi pengguna dilakukan dengan *Username* dan *Password*. *Password* disimpan dalam basis data dalam bentuk *Hash*. Pengguna yang sudah *Login* menggunakan *Username* dan *Password*. Setelah itu, diberikan sebuah *Token* untuk mengakses aplikasi tersebut. *Token* ini disusun dengan menggunakan *JSON Web Token* (JWT). JWT yang telah diberikan harus disertakan dalam *Authorization Request Header* yang dikirimkan oleh *Client* agar dapat memperoleh informasi yang mengharuskan otentikasi.
Aplikasi ini menyediakan informasi berdasarkan API yang diminta oleh *client*. Informasi tersebut disajikan dalam format *Javascript Object Notation* (JSON). Tabel 1 menunjukkan API apa saja yang dibutuhkan pada aplikasi,

informasi yang didapat dari API tersebut, beserta keterangan-keterangannya.

Tabel 1. API dalam Aplikasi

Routes	URL	Tujuan	Keterangan
GET	/api/	Mengetahui status aplikasi	Balasan yang diterima dari server adalah "OK"
POST	/api/rMovie/user/login	Melakukan Login	Client mengirimkan request yang berisi fields yakni username dan password. Jika sukses maka server mengirimkan token JWT sebagai balasan.
GET	/api/rMovie/review/:id	Mendapatkan review dari suatu film.	{:id} adalah kode identifikasi film dalam sistem. {:id} diperoleh melalui /api/rMovie
GET	/api/rMovie/user/review	Mendapatkan semua review yang telah dibuat user.	Header berisi token JWT yang didapatkan dari login.
GET	/api/rMovie/	Mendapatkan semua film	
GET	/api/rMovie/info/:id	Mendapatkan info dari suatu film	{:id} adalah kode identifikasi film dalam sistem. {:id} diperoleh melalui /api/rMovie
GET	/api/rMovie/allCategory	Mendapatkan semua kategori dalam film	
GET	/api/rMovie/:category_id	Mendapatkan semua film dalam berdasarkan kategori film	{:category_id} adalah kode identifikasi kategori film dalam sistem. {:category_id} diperoleh melalui /api/rMovie/allCategory

Perangkat sistem pengelolaan basis data yang digunakan untuk ke 2 aplikasi tersebut adalah Mysql versi 8.0.31. Aplikasi yang dibangun untuk Golang dibangun di atas Golang versi 1.19.2. Aplikasi tersebut dibangun dengan menggunakan *framework* Echo versi 4.9.1. JWT dalam aplikasi ini

diimplementasikan dengan library golang-jwt versi 4.9.1. Algoritma Bcrypt dalam aplikasi ini diimplementasi dengan modul Crypto/Bcrypt versi 0.2.0. Aplikasi Node.js dibangun pada Node versi 16.17.1 dengan menggunakan *framework* Express versi 4.18.2. Otentikasi JWT ditangani oleh modul jsonwebtoken versi 8.5.1 dan Bcrypt versi 5.1.0.

b) *Sprints*, pada proses ini memulai untuk membentuk dan membangun API yang telah direncanakan pada proses *Backlog*. Jika ada tambahan API pada proses ini tidak bisa ditambahkan sehingga jika ada perubahan pada fitur maka harus melalui proses awal.
c) *Demos*, pada proses ini dilakukan uji coba dari API tersebut agar dapat mengetahui apakah API tersebut dapat digunakan dengan baik atau tidak.

Penulis menggunakan 2 metode pada penelitian ini yaitu yang pertama metode penerapan di mana penulis sekaligus peneliti membuat beberapa API yang dibangun menggunakan bahasa pemrograman Golang dan Node.js. Di mana API tersebut digunakan dalam pengujian. Dalam membangun API dibantu dengan menggunakan metode *Scrum*, rancangan apa saja yang dibuat API pada penelitian ini ada pada proses *Backlog*, kemudian dilanjut dengan proses *Sprint* di mana pada tahap ini peneliti membangun API sesuai dengan apa yang telah ditetapkan sebelumnya. Pada proses terakhir yaitu melakukan demo pada API yang telah dibuat.

Metode kedua adalah metode eksperimen pada API yang telah dibuat pada proses metode penerapan. Setiap jenis API memiliki masing-masing versi bahasa pemrograman Golang dan Node.js. Pada tahap ini dilakukan pengujian dalam perbandingan antar bahasa pemrograman khusus nya pada sisi *Backend*. Penelitian ini dilaksanakan dengan 2 unit komputer. Jenis komputer yang pertama digunakan sebagai menjalankan aplikasi atau sisi *server* memiliki CPU Intel Core i5-8265U, RAM 8GB, serta sistem operasi Ubuntu 20.01 LTS. Sedangkan jenis komputer yang digunakan sebagai pengujian atau sisi *client* memiliki CPU Intel Core i3, RAM 4GB, serta sistem operasi Windows 7.

Pengukuran kecepatan respon dilakukan dengan menggunakan JMeter. JMeter dapat dikonfigurasikan untuk mengirimkan berbagai tipe, volume, pengaturan waktu, serta isi request. JMeter juga dapat menyajikan hasil pengujian tabel dan grafik. Dalam penelitian ini, JMeter dikonfigurasikan untuk mengirimkan 10000 request pada saat yang bersamaan. Matrix yang digunakan pada pengukuran kecepatan respon adalah milidetik [7].

Pengukuran konsumsi CPU dan memori dilakukan dengan menggunakan program bawaan dari sistem operasi Linux yaitu PS. Program tersebut dieksekusi setiap 0,5 detik dan outputnya disaring dengan program *grep* sehingga hanya menampilkan proses yang relevan serta disalurkan ke dalam sebuah berkas teks untuk kemudian analisis. Matrix yang digunakan pada pengukuran CPU dan memori adalah berapa % (persen) penggunaannya [5].

Cara melakukan pengujian tersebut dalam pengambilan data penelitian dengan prosedur untuk tiap-tiap URL dalam masing-masing aplikasi bahasa pemrograman golang dan Node.js. Pertama aplikasi dijalankan hingga siap

digunakan. Program ps dijalankan, kemudian aplikasi JMeter dijalankan pada komputer dan skenario pengujian dieksekusi. Aplikasi dihentikan dan komputer dimatikan. Kemudian dihidupkan untuk API selanjutnya hingga semua API dari setiap bahasa pemrograman Golang dan Node.js selesai dilakukan pengujian.

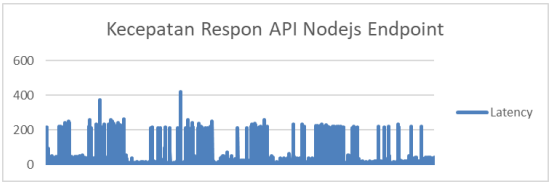
Saat melakukan analisis data empiris dari hasil pengujian yang telah didapatkan dengan matrix yang menjadi patokan dalam perbandingan tersebut. Hasil yang didapatkan tersebut kemudian dilakukan komparasi atau perbandingan dari kedua bahasa pemrograman yaitu Golang dan Node.js pada tiap-tiap jenis API yang sama.

3. HASIL DAN PEMBAHASAN

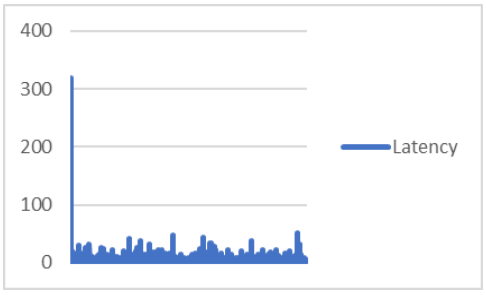
a) *Status Aplikasi*

Query dalam status aplikasi yang terdapat pada URL */api/* adalah tugas sederhana tanpa perlu mengakses basis data maupun melakukan algoritma kriptografi yang rumit. *Request* terhadap URL ini diselesaikan oleh Node.js dengan rata-rata durasi 8,087 milidetik (Gambar 2). Sedangkan Golang memberikan respon dalam rata-rata 3,4455 milidetik (Gambar 3).

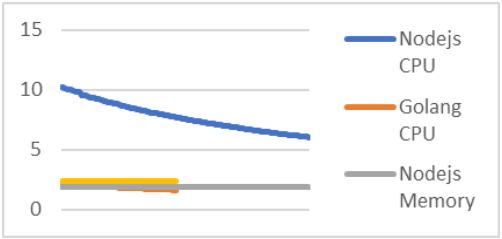
Dalam rangka merespon *request* mengenai status aplikasi, Node.js memakan rata-rata 7,761413% CPU dan 1,9% memori. Golang memakai rata-rata 1,853488% CPU dan 2,4% memori (Gambar 4).



Gambar 2. Waktu Respon Node.js URL */api/*



Gambar 3. Waktu Respon Node.js URL */api/*



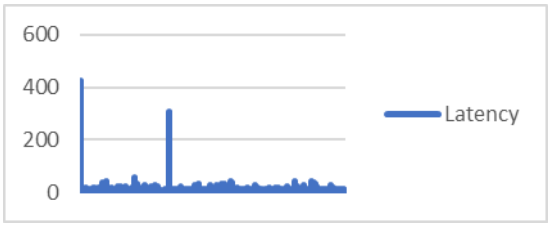
Gambar 4. CPU dan Memori URL */api/*

Jurnal Riset Sistem Informasi Dan Teknik Informatika (JURASIK)
Volume 8, Nomor 1, Februari 2023, pp 155-168
Terakreditasi Nomor 204/E/KPT/2022 | ISSN: 2527-5771/EISSN: 2549-7839
https://tunasbangsa.ac.id/ejurnal/index.php/jurasik

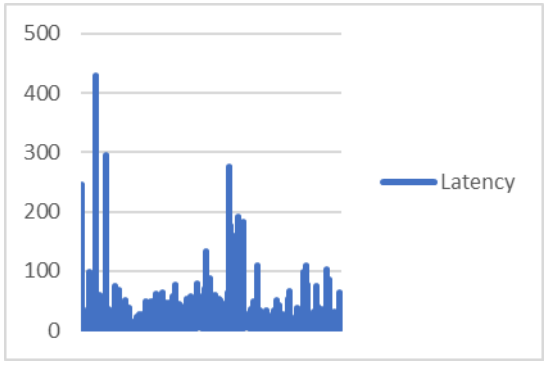
b) Daftar Review Dari Suatu Film

API *Daftar Review Dari Suatu Film* merupakan salah satu fitur dari aplikasi yang hanya bisa diakses dengan memasukan otentikasi dan pengambilan data berdasarkan film yang dipilih. Dengan kata lain untuk dapat mengakses fitur tersebut harus melakukan *Login* terlebih dahulu. Golang memberikan respon terhadap *request* dari *Client* dengan rata-rata 5,6465 milidetik (Gambar 5). Sedangkan Node.js mendapatkan respon rata-rata 9,2555 milidetik (Gambar 6).

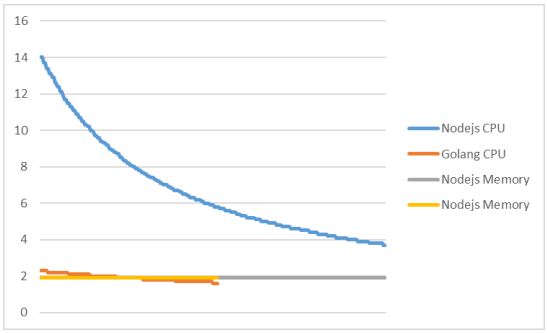
Dalam rangka merespon *request* mengenai daftar review dari suatu film, Golang mengonsumsi rata-rata 1,9216% CPU dan 1,9% memori. Node.js mengonsumsi rata-rata 6,732099% CPU dan 1,9% memori (gambar 7).



Gambar 5. Waktu Respon Golang untuk URL */api/rMovie/review/:id*



Gambar 6. Waktu Respon Node.js untuk URL */api/rMovie/review/:id*



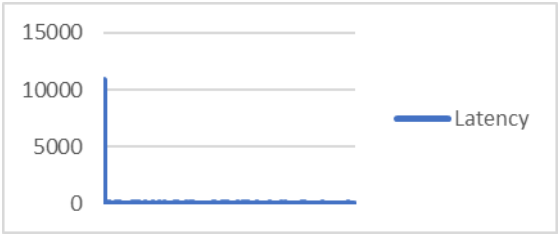
Gambar 7. Penggunaan Sumber Daya untuk URL */api/rMovie/review/:id*

c) *Daftar Review dari User*

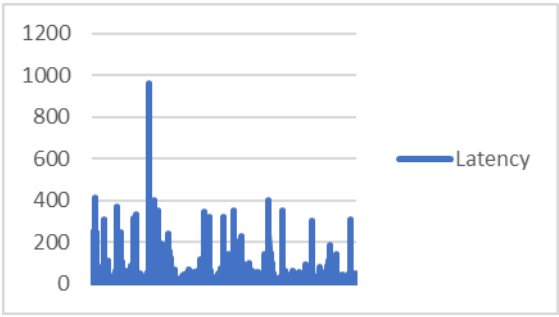
API *Daftar Review dari User* melibatkan penggunaan verifikasi JWT untuk mendapatkan data user dan pengambilan data dari database review berdasarkan *User* tersebut. Golang memberikan respon rata-rata dalam 7,0102 milidetik

(Gambar 8). Kemudian Node.js mendapatkan respon rata-rata dalam waktu 12,8376 milidetik (Gambar 9).

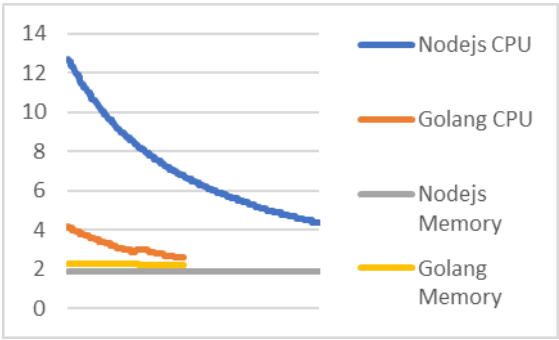
Untuk URL */api/rMovie/user/review/* Golang mengonsumsi rata-rata 3,181457% dan 2,261589% memori. Sementara Node.js mengonsumsi rata-rata 7,110061% CPU dan 1,9% memori (Gambar 10).



Gambar 8. Waktu Respon Golang untuk URL */api/rMovie/user/review/*



Gambar 9. Waktu Respon Node.js untuk URL /api/rMovie/user/review/



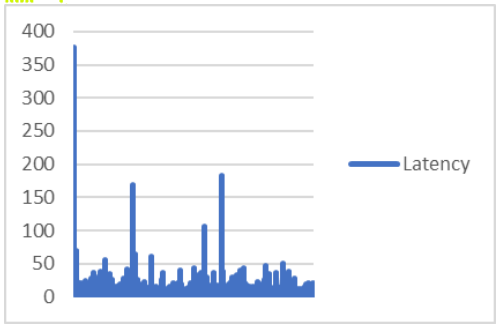
Gambar 10. Penggunaan Sumber Daya untuk URL */api/rMovie/user/review/*

d) *Daftar Film*

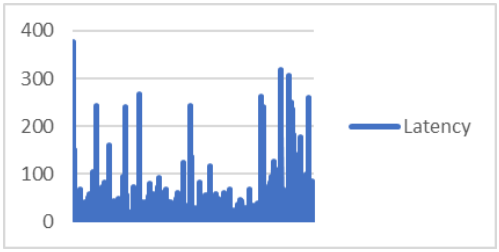
Daftar Film, seperti halnya daftar film yang ada pada aplikasi tersebut. Pada API ini tidak perlu dari sisi *Client* untuk menyertakan JWT dalam header. Golang memberikan respon dalam waktu rata-rata 4,9431 milidetik (Gambar 11). Sedangkan rata-rata waktu respon Node.js adalah 8,5652 milidetik (Gambar 12).

Dari segi konsumsi sumber daya, Golang mengonsumsi rata-rata 2,65137615% CPU dan 2,2266055 memori. Node.js mengonsumsi 6,36681614% CPU dan 1,9 memori (Gambar 13).

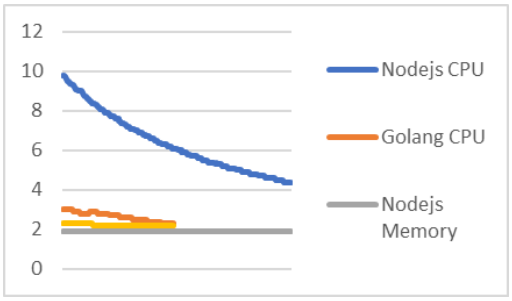
Jurnal Riset Sistem Informasi Dan Teknik Informatika (JURASIK)
Volume 8, Nomor 1, Februari 2023, pp 155-168
Terakreditasi Nomor 204/E/KPT/2022 | ISSN: 2527-5771/EISSN: 2549-7839
https://tunasbangsa.ac.id/ejurnal/index.php/jurasik



Gambar 11. Waktu Respon Golang untuk URL */api/rMovie/*



Gambar 12. Waktu Respon Node.js untuk URL */api/rMovie/*

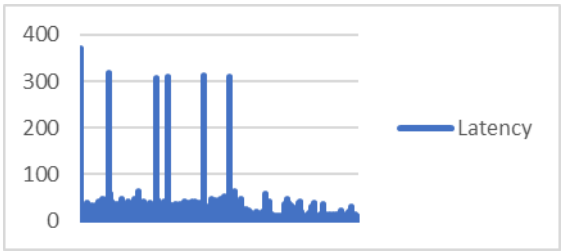


Gambar 13. Penggunaan Sumber Daya untuk URL */api/rMovie/*

e) *Detail Film*

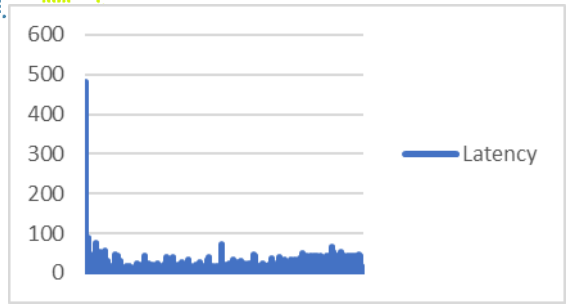
API terhadap detail sebuah film tidak melibatkan verifikasi JWT. Golang memberikan respon dalam waktu rata-rata 5,9535 milidetik (Gambar 14). Node.js memberi respon dalam waktu rata-rata 6,8335 milidetik (Gambar 15).

Untuk menyajikan informasi mendetail mengenai suatu film yang spesifik, Golang mengonsumsi rata-rata 2,700787% CPU dan 2,23622% memori. Node.js mengonsumsi 11,61141% CPU dan 2% memori (Gambar 16).

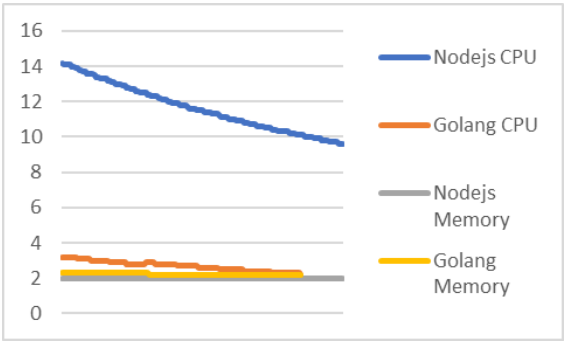


Gambar 14. Waktu Respon Golang untuk URL */api/rMovie/info/:id*

Jurnal Riset Sistem Informasi Dan Teknik Informatika (JURASIK)
Volume 8, Nomor 1, Februari 2023, pp 155-168
Terakreditasi Nomor 204/E/KPT/2022 | ISSN: 2527-5771/EISSN: 2549-7839
https://tunasbangsa.ac.id/ejurnal/index.php/jurasik



Gambar 15. Waktu Respon Node.js untuk URL */api/rMovie/info/:id*

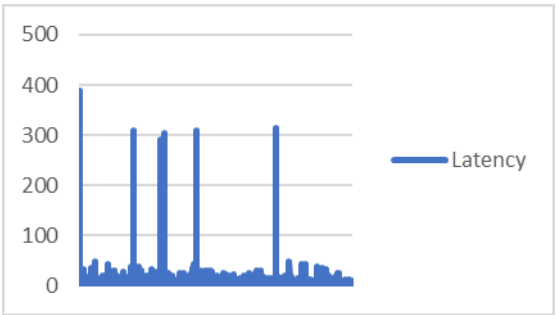


Gambar 16. Penggunaan Sumber Daya untuk URL */api/rMovie/info/:id*

f) *Daftar Kategori*

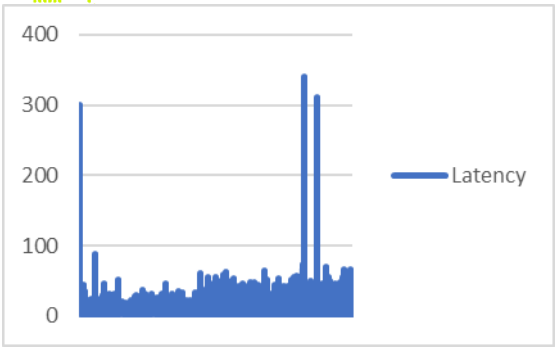
Daftar Kategori merupakan salah satu API yang tidak mewajibkan pengguna untuk *Login* terlebih dahulu, dan demikian tidak melibatkan perhitungan kriptografi, baik untuk memverifikasi JWT maupun membandingkan *hash*. Tugas ini melibatkan akses ke basis data. Waktu respon untuk mengakses API ini pada Golang rata-rata adalah 4,5588 milidetik (Gambar 17). Sedangkan pada Node.js memberikan respon dalam waktu rata-rata 7,3349 (Gambar 18).

Untuk menampilkan informasi daftar kategori film Golang mengonsumsi rata-rata 3,197059% CPU dan 2,194118% memori. Node.js mengonsumsi 7,332143% CPU dan 2% memori (Gambar 19).

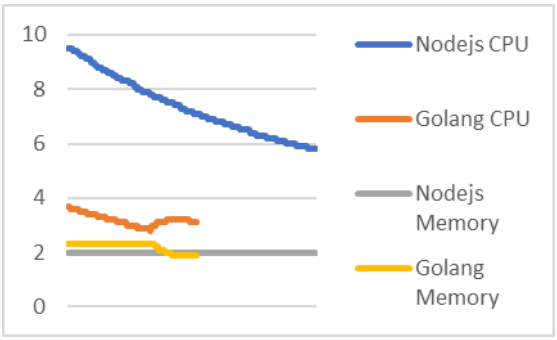


Gambar 17. Waktu Respon Golang untuk URL */api/rMovie/allCategory/*

Jurnal Riset Sistem Informasi Dan Teknik Informatika (JURASIK)
Volume 8, Nomor 1, Februari 2023, pp 155-168
Terakreditasi Nomor 204/E/KPT/2022 | ISSN: 2527-5771/EISSN: 2549-7839
https://tunasbangsa.ac.id/ejurnal/index.php/jurasik



Gambar 18. Waktu Respon Node.js untuk URL */api/rMovie/allCategory/*

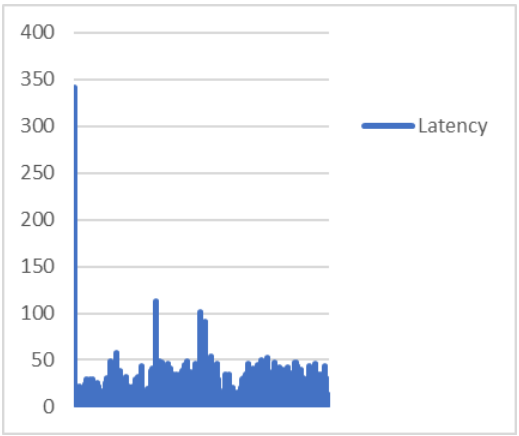


Gambar 19. Penggunaan Sumber Daya untuk URL */api/rMovie/allCategory/*

g) *Daftar Film Berdasarkan Kategori*

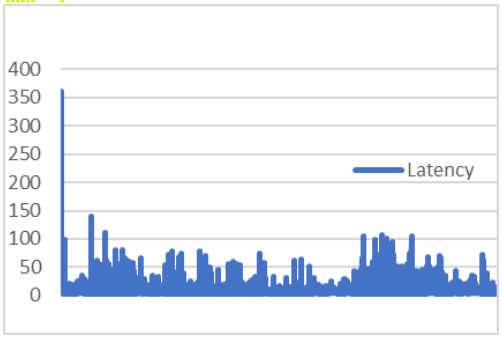
Daftar film berdasarkan kategori merupakan API yang tidak perlu penggunaannya untuk *Login* terlebih dahulu. Dalam hal ini API hanya melakukan basis data dan mengambil data film berdasarkan kategori dari id kategori yang telah diinputkan diambil dari API daftar kategori. Waktu respon bagi Golang rata-rata adalah 6,61635 milidetik (Gambar 20). Sedangkan untuk Node.js memerlukan waktu respon rata-rata 8,3886 milidetik (Gambar 21).

Untuk memperlihatkan informasi film berdasarkan kategorinya, maka dihasilkan Golang mengonsumsi rata-rata 2,731061% CPU dan 2,3 memori (Gambar 22).

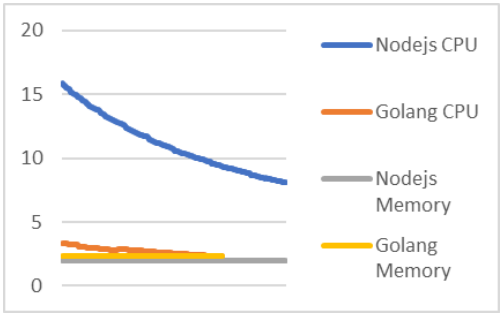


Gambar 20. Waktu Respon Golang untuk URL */api/rMovie/:id*

Jurnal Riset Sistem Informasi Dan Teknik Informatika (JURASIK)
Volume 8, Nomor 1, Februari 2023, pp 155-168
Terakreditasi Nomor 204/E/KPT/2022 | ISSN: 2527-5771/EISSN: 2549-7839
https://tunasbangsa.ac.id/ejurnal/index.php/jurasik



Gambar 21. Waktu Respon Node.js untuk URL */api/rMovie/:id*

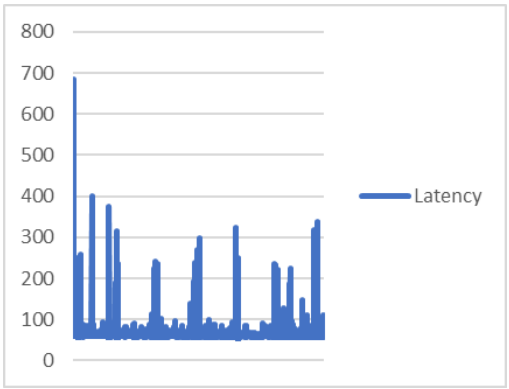


Gambar 22. Penggunaan Sumber Daya untuk URL */api/rMovie/:id*

h) *Login*

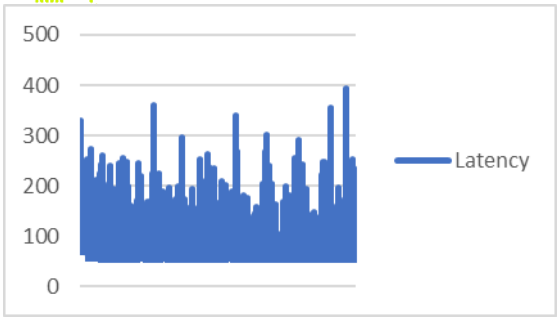
Proses *Login* melibatkan akses basis data sekaligus eksekusi algoritma *Bcrypt* untuk membandingkan *hash* password. *Bcrypt* adalah nama algoritma kriptografi yang relatif lebih rumit dibandingkan kriptografi untuk verifikasi JWT dan sebagai *hashing* yang memiliki banyak iterasinya secara bertambah terus menerus untuk memperlambat dan bertahan dari serangan *brute force* [8]. Golang memberikan dalam waktu rata-rata 62,7 milidetik (Gambar 23). Kemudian pada Node.js memerlukan rata-rata waktu respon 79,2829 milidetik (Gambar 24).

Untuk melakukan login Golang mengonsumsi sumber daya dengan rata-rata 1,15032% CPU dan 2,002985% memori. Node.js mengonsumsi 4,710952% CPU dan 1,9% memori (Gambar 25).

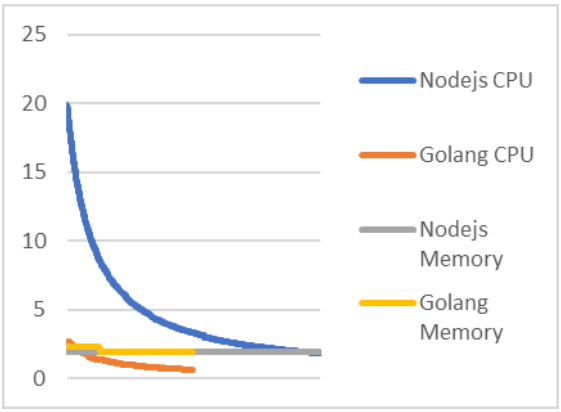


Gambar 23. Waktu Respon Golang untuk URL */api/rMovie/user/login*

Jurnal Riset Sistem Informasi Dan Teknik Informatika (JURASIK)
Volume 8, Nomor 1, Februari 2023, pp 155-168
Terakreditasi Nomor 204/E/KPT/2022 | ISSN: 2527-5771/EISSN: 2549-7839
https://tunasbangsa.ac.id/ejurnal/index.php/jurasik



Gambar 24. Waktu Respon Node.js untuk URL */api/rMovie/user/login*



Gambar 25. Penggunaan Sumber Daya untuk URL */api/rMovie/user/login*

4. SIMPULAN

Berdasarkan dari hasil penelitian ini, maka dapat ditarik kesimpulan bahwa dalam melakukan penelitian, kedua bahasa pemrograman Golang dan Node.js dapat berjalan baik hingga menyelesaikan tugas-tugas yang diberikan. Dari kedua objek penelitian, Golang adalah bahasa pemrograman yang memiliki waktu respon lebih cepat dibanding Node.js dalam melaksanakan tugas pengambilan data sederhana, melibatkan perhitungan-perhitungan kriptografi yang intensif. Dalam menyediakan respon langsung terhadap *request* yang diterima. Dalam penggunaan sumber daya Golang memiliki penggunaan CPU lebih sedikit dibanding Node.js. Node.js memiliki keseimbangan yang baik dalam penggunaan sumberdaya khususnya pada memori.

DAFTAR PUSTAKA

[1] A. Mubariz, D. Nur, E. Tungadi, and M. N. Y. Utomo, "Perancangan Back-End Server Menggunakan Arsitektur Rest dan Platform Node . JS (Studi Kasus : Sistem Pendaftaran Ujian Masuk Politeknik Negeri Ujung Pandang)," *Semin. Nas. Tek. Elektro dan Inform.*, pp. 72–77, 2020.

[2] N. Agung, *Dasar Pemrograman Golang*. 2021.

[3] "Insights 2020 Developer Survey," *Stack Overflow*, 2020. https://insights.stackoverflow.com/survey/2020#technology (accessed Nov. 23, 2021).

[4] A. Firdaus, S. Widodo, A. Sutrisman, S. G. Fadhilah Nasution, and R. Mardiana, "Rancang Bangun Sistem Informasi Perpustakaan Menggunakan WEB Sevice Pada

Jurusan Teknik Komputer Polsri," *J. Inform.*, vol. 5, no. 2, pp. 81–87, 2019.

[5] A. C. Rompis and R. F. Aji, "Perbandingan Performa Kinerja Node.js, PHP, dan Python dalam Aplikasi REST," *CogITo Smart J.*, vol. 4, no. 1, p. 171, 2018, doi: 10.31154/cogito.v4i1.92.171-187.

[6] R. Gutama and T. Dirgahayu, "Implementasi Scrum Pada Manajemen Proyek Pengembangan Aplikasi Sistem Monitoring dan Evaluasi Pembangunan (SMEP)," *Informatics Dep. Univ. Islam Indones.*, vol. Vol 2, p. 7, 2021.

[7] A. Ridwan, "Analisis Perbandingan Performa Apache Web Server Dan Nginx Menggunakan Apache Jmeter," *J. Teknoif ITP*, vol. 8, no. 2, pp. 87–92, 2020, doi: 10.21063/jtif.2020.v8.2.87-92.

[8] G. D. M. Zulma, H. B. Seta, and T. Yuniati, "Implementasi Algoritma Aes Dan Bcrypt Untuk Pengamanan File Dokumen," *Inform. J. Ilmu Komput.*, vol. 18, no. 2, p. 163, 2022, doi: 10.52958/iftk.v18i2.4667.