Analisis Perbandingan Kinerja Backend API Menggunakan PHP, Golang, dan JavaScript

Performance Analysis of Backend API Using PHP, Golang, and Node JS

Fanes Pratama[1], Ahmad Farisi[2]
[1,2]Sistem Informasi, Universitas Multi Data Palembang
E-mail: [1]fanes23.pratama@mhs.mdp.ac.id, [2]ahmadfarisi@mdp.ac.id

Abstrak

Pengembangan backend API yang efisien sangat penting dalam mendukung aplikasi web modern. Namun, pemilihan bahasa pemrograman dan metode *query* yang optimal masih menjadi tantangan bagi pengembang. Penelitian ini bertujuan untuk membandingkan kinerja backend RESTful API yang dibangun menggunakan tiga bahasa pemrograman (Go, PHP, dan JavaScript) serta empat metode pengambilan data (Raw SQL, ORM, Query Builder, dan Stored Procedure). Metode penelitian yang digunakan adalah kuantitatif *true-experimental*, dengan pengujian Load Testing, Spike Testing, dan Stress Testing untuk mengevaluasi jumlah permintaan yang berhasil, penggunaan CPU, dan penggunaan memori. Hasil pengujian menunjukkan bahwa Go dengan Raw SQL memiliki kinerja tertinggi dalam jumlah permintaan, waktu respons, dan penanganan beban, diikuti oleh Node.js, sementara PHP memiliki kinerja terendah.

Kata kunci: Backend API, Pengujian Kinerja, Metode Query

Abstract

Efficient backend API development is crucial for supporting modern web applications. However, selecting the optimal programming language and query method remains a challenge for developers. This study aims to compare the performance of RESTful API backends built using three programming languages (Go, PHP, and JavaScript) and four data retrieval methods (Raw SQL, ORM, Query Builder, and Stored Procedure). The research method employed is quantitative true experimental, utilizing Load Testing, Spike Testing, and Stress Testing to evaluate successful request rates, CPU usage, and memory consumption. The test results show that Go with Raw SQL delivers the highest performance in terms of request handling, response time, and load management, followed by Node.js, while PHP has the lowest performance.

Keywords: Backend API, Performance Testing, Query Method

1. PENDAHULUAN

Pengembangan web umumnya melibatkan sisi klien dan sisi server, yang disebut sebagai pengembangan web *full-stack*. Pengembangan *full-stack* telah mengalami pertumbuhan yang signifikan dalam beberapa tahun terakhir seiring dengan meningkatnya kebutuhan untuk pengembangan web dengan pertumbuhan internet dan *e-commerce*. Dengan munculnya *cloud*, arsitektur *micro-services*, dan kebutuhan untuk membuat serta memelihara aplikasi web yang kompleks, pengembang *full-stack* semakin diminati [1].

REST (*Representational State Transfer*) adalah arsitektur yang digunakan untuk merancang layanan yang dikonsumsi di berbagai platform dan lingkungan untuk mendukung interoperabilitas dan WWW (*World Wide Web*). *Statelessness* dan kesiapan konsumsi lintas platform adalah dua atribut utama dari arsitektur ini [2]. Saat ini, RESTful API telah menjadi standar yang umum digunakan dalam pengembangan sistem yang memerlukan penyediaan API untuk pihak eksternal [3].

Berbagai bahasa pemrograman telah berkembang pesat dan digunakan secara luas dalam pengembangan aplikasi dengan arsitektur RESTful API, seperti PHP, Go, dan JavaScript. Bahasa

pemrograman PHP adalah salah satu bahasa yang didukung oleh sebagian besar server saat ini, sehingga mengembangkan aplikasi berbasis web menggunakan PHP akan menjadi lebih mudah bagi para *programmer* [4] Golang (*Go language*) adalah bahasa pemrograman *open-source* yang dikembangkan oleh Google, yang memiliki keunggulan dalam mendukung *concurrency*, memiliki sistem *garbage collection* yang efisien, serta cepat dan andal dalam skala besar, menjadikannya pilihan ideal untuk aplikasi skala besar dengan arsitektur yang bersih [5]. Node.js, lingkungan *runtime* JavaScript, sangat populer di kalangan pengembang *full-stack* untuk membangun aplikasi web. Ini memungkinkan pengembang menulis kode sisi server dalam bahasa pemrograman JavaScript dan mengelola sisi klien dan server sekaligus [6].

Saat membangun sebuah aplikasi, pengembang perlu mengetahui teknologi yang tepat untuk digunakan. Studi yang dilakukan [7] menyebutkan bahwa Node.js memiliki kinerja yang lebih baik dibandingkan PHP dalam hal menangani permintaan data. Dalam hal kinerja aplikasi, penambahan *resource server* juga tidak selalu memberikan dampak signifikan terhadap peningkatan kinerja sistem tanpa adanya pengaturan tambahan pada teknologi yang digunakan. Hal tersebut dibuktikan melalui penelitian yang dilakukan oleh [8] yaitu dengan menambahkan core CPU dan memori RAM, lalu membandingkan hasil tes sebelum dan sesudah penambahan *resource server*.

Pengujian kinerja merupakan proses untuk mengevaluasi kondisi perangkat lunak saat menghadapi beban kerja tertentu [9]. Untuk melakukan pengujian kinerja aplikasi yang berfokus pada *input* dan *output* serta *resource server* yang digunakan, ada beberapa metode yang sering digunakan, salah satunya adalah Blackbox Testing, yang merupakan pengujian fungsional perangkat lunak tanpa mengetahui struktur internal program. Metode pengujian ini berfokus pada spesifikasi fungsional perangkat lunak, serangkaian kondisi input, dan melakukan pengujian pada fungsionalitas program [10].

Penelitian yang menerapkan metode Blackbox Testing, seperti dalam studi oleh [11] menyebutkan bahwa metode Raw SQL memiliki kinerja yang lebih baik dibandingkan *Object-Relational Mapping* (ORM) dalam menangani data besar. Studi lain oleh [12] juga menyebutkan bahwa metode Stored Procedure memiliki kinerja yang lebih baik daripada Function dalam hal penyimpanan data ke dalam database. Dari dua penelitian tersebut, dapat ditarik informasi bahwa selain bahasa pemrograman, metode pengambilan data (*query*) juga dapat berdampak signifikan terhadap kinerja aplikasi. Oleh karena itu, pemilihan metode *query* yang tepat sangat penting untuk mengoptimalkan efisiensi dan kecepatan aplikasi, terutama dalam skala yang besar.

Penelitian terdahulu lainnya, seperti yang dilakukan [13] telah menerapkan metode Load Testing untuk mengetahui batasan data dan permintaan yang dapat diproses oleh aplikasi. Namun, penelitian ini tidak melakukan pengujian lain seperti Spike Test, Stress Test, dan Soak Test seperti yang dilakukan [1]. Pengujian yang beragam sangat diperlukan untuk menguji kinerja aplikasi secara lebih detail di berbagai kondisi, seperti lonjakan tiba-tiba pada jumlah pengguna (*spike*), aplikasi berada di bawah tekanan ekstrem (*stress*), atau aplikasi diuji untuk jangka waktu yang lama dengan beban stabil (*soak*). Pentingnya menerapkan metode pengujian yang beragam juga didukung studi oleh [14], yang menyebutkan bahwa satu kali request tidak cukup untuk mengukur penggunaan memori secara akurat, meskipun jumlah data yang digunakan dalam pengujian mencapai 100.000.

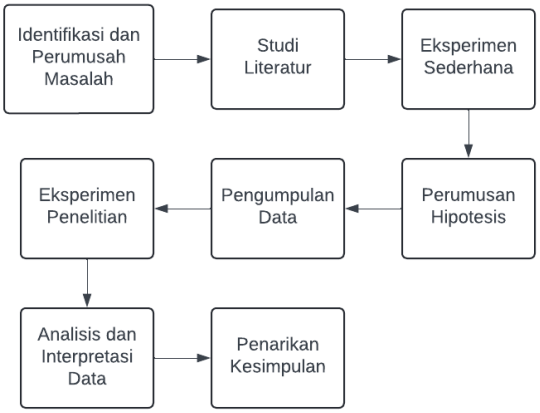
Untuk membuktikan secara lebih rinci hasil dari beberapa penelitian sebelumnya yang menyatakan adanya perbedaan kinerja antara bahasa pemrograman dan metode *query* yang berbeda, penelitian ini dilakukan dengan cakupan parameter yang lebih luas. Studi ini membandingkan tiga bahasa pemrograman yaitu JavaScript (Node.js), PHP, dan Go, serta empat metode *query* yang berbeda, yaitu ORM, Raw SQL, Stored Procedure, dan Query Builder. Selain itu, penelitian ini juga menerapkan tiga metode pengujian, yaitu Stress Testing, Load Testing, dan Spike Testing, guna mendapatkan analisis yang lebih komprehensif terhadap kinerja sistem.

2. METODE PENELITIAN

Penelitian ini menggunakan metode kuantitatif *true-experimental* untuk menentukan kombinasi bahasa pemrograman dan metode *query* yang paling optimal dalam meningkatkan kinerja backend RESTful API. Metode ini dipilih karena memungkinkan kontrol ketat terhadap variabel-variabel penelitian dan memungkinkan penarikan kesimpulan mengenai pengaruh kombinasi yang diuji terhadap metrik kinerja, seperti jumlah permintaan yang berhasil, penggunaan CPU, dan penggunaan memori.

2.1 Tahapan Penelitian

Tahapan dari penelitian ini dapat dilihat pada Gambar 1 berikut.



Gambar 1 Tahapan Penelitian

2.1.1 Identifikasi dan Perumusan Masalah

Pada tahap ini, mengidentifikasi masalah seperti memilih bahasa pemrograman dan metode *query* yang tepat dapat mempengaruhi kinerja dan sumber daya aplikasi. Sebagian besar penelitian terdahulu hanya membandingkan kinerja dari beberapa kombinasi bahasa pemrograman dan metode *query* dalam kondisi tertentu. Oleh karena itu, penelitian ini difokuskan untuk mengidentifikasi kombinasi yang mampu memberikan kinerja yang optimal dalam hal jumlah permintaan yang berhasil diproses, penggunaan CPU, dan penggunaan memori.

2.1.2 Studi Literatur

Penelitian ini didasarkan pada studi-studi terdahulu yang diperoleh dari artikel jurnal, buku, dan artikel daring dari sumber-sumber terpercaya. Studi-studi ini mendukung dalam mengidentifikasi bahasa pemrograman yang populer digunakan dalam pengembangan aplikasi backend saat ini, *framework* dan metode *query* yang sering diterapkan, serta metode pengujian yang digunakan untuk mendapatkan hasil yang relevan dan valid untuk masalah yang telah dirumuskan pada tahap sebelumnya.

2.1.3 Eksperimen Sederhana

Penelitian ini melibatkan eksperimen sederhana dengan menggunakan data dan tabel *dummy* untuk memperoleh hipotesis awal seperti yang dilakukan pada latar belakang penelitian. Hasil dari eksperimen sederhana ini akan diinterpretasikan sebagai dasar untuk penelitian utama yang akan dilakukan. Pendekatan ini bertujuan untuk memberikan pemahaman awal yang akan mendukung analisis lebih lanjut dalam konteks penelitian yang sesungguhnya.

2.1.4 Perumusan Hipotesis

Pada tahap perumusan hipotesis, penelitian ini menetapkan dugaan awal berdasarkan studi literatur dan eksperimen sederhana. Hipotesis yang diajukan mencakup perbedaan kinerja

antara kombinasi bahasa pemrograman, metode *query*, dan metode pengujian dalam memproses permintaan, serta dalam penggunaan CPU dan memori. Hasil dari hipotesis awal ini akan dievaluasi dan divalidasi setelah dilakukan penelitian utama.

2.1.5 Pengumpulan Data

Penelitian ini mengumpulkan data berupa *source code* backend aplikasi SIRAT Mobile yang digunakan oleh PT Goenawan Erawisata Cabang Palembang. Data yang diperoleh mencakup bahasa pemrograman, struktur tabel database, dan metode *query* yang diterapkan. Data ini digunakan sebagai referensi dalam studi kasus penelitian. Namun, data tersebut tidak secara langsung memengaruhi tujuan utama penelitian, sehingga kesimpulan yang dihasilkan tetap berfokus pada analisis konseptual yang lebih luas.

2.1.6 Eksperimen Penelitian

Penelitian ini menggunakan desain eksperimen faktorial, dengan tiga variabel bebas (bahasa pemrograman, metode *query*, dan metode pengujian) dan tiga variabel terikat (jumlah permintaan berhasil, penggunaan CPU, dan penggunaan memori). Desain ini dipilih untuk mengevaluasi efek kombinasi antara bahasa pemrograman dan metode *query* terhadap kinerja aplikasi backend API. Alur penelitian secara detail akan dijelaskan pada sub-bab 2.4.

2.1.7 Analisis dan Interpretasi Data

Data yang diperoleh dari pengujian akan diproyeksikan menggunakan *library* Matplotlib ke dalam bentuk *bar chart* dan *line chart* terpisah untuk setiap parameter dan metode pengujian. Setiap *chart* akan dianalisis untuk membuktikan hipotesis penelitian yang telah dirumuskan sebelumnya.

2.1.8 Penarikan Kesimpulan

Berdasarkan hasil penelitian, analisis data, dan interpretasi data yang telah dilakukan, selanjutnya akan dihasilkan beberapa kesimpulan yang akan menjawab rumusan masalah serta mencapai tujuan yang telah ditetapkan.

2.2 Ruang Lingkup Penelitian

Penelitian ini membandingkan tiga bahasa pemrograman, yaitu PHP, Go, dan JavaScript (Node.js), dengan menguji berbagai metode *query*, seperti Raw SQL, ORM, Query Builder, dan Stored Procedure. Pengujian kinerja dilakukan menggunakan metode Load Testing, Spike Testing, dan Stress Testing dengan *software* k6.io. Data *dummy* yang digunakan terdiri dari 50 baris data pada tabel *jamaah* (*id, id_paket, nama_paspor, tipe_kamar, no_hp, kode_keluarga,* dan *urutan*) yang di-*join* dengan tabel *paket* (*id, nama, harga_quad, harga_triple, harga_double,* dan *roomlist*) dan tabel *fasilitas* (*id, id_jamaah, nama_fasilitas,* dan *deskripsi*). Pengujian dilakukan pada komputer dengan spesifikasi CPU Intel(R) Core(TM) i5-6500T CPU @ 2.50GHz (4 *core*), Memori RAM 4GB DDR3, dan Sistem Operasi Linux Ubuntu Server versi 22.04.3 LTS. Backend API diimplementasikan menggunakan kode native dari masing-masing bahasa tanpa *framework* HTTP tambahan, dengan pengujian melalui metode HTTP GET. Analisis meliputi pengukuran sumber daya penggunaan CPU dan memori menggunakan Glances, visualisasi data dengan Matplotlib, serta interpretasi hasil pengujian tanpa analisis statistik khusus.

2.3 Framework Eksperimen

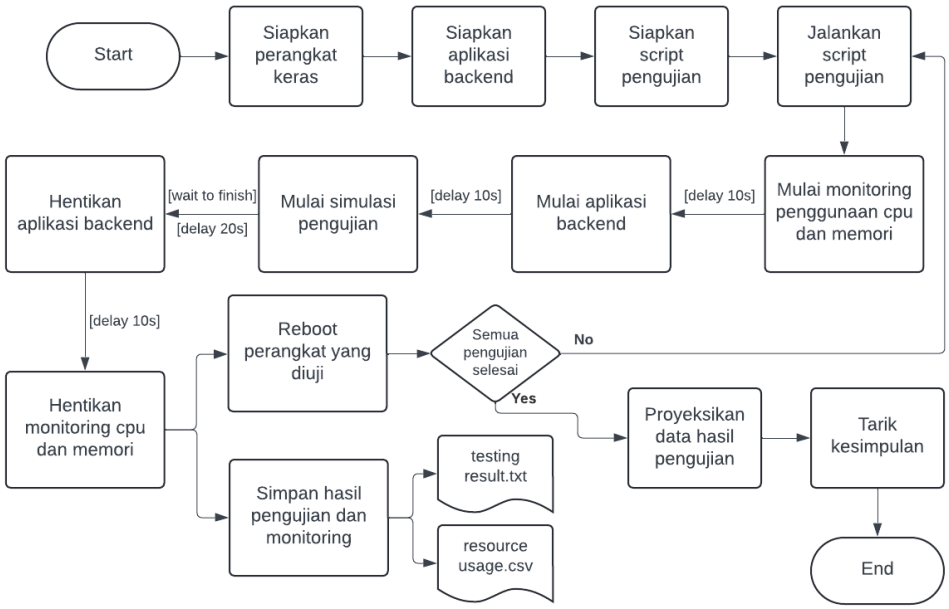
Framework-framework yang digunakan untuk setiap kombinasi bahasa pemrograman dan metode *query* dalam eksperimen ini dapat dilihat pada Tabel 1 berikut.

Tabel 1 Framework Eksperimen

Metode Query	Go	JavaScript	PHP
ORM	gorm	sequelize	eloquent
Query Builder	squirrel	knex	capsule
Raw SQL	go-sql-driver	mysql2	native
Stored Procedure	go-sql-driver	mysql2	native

2.4 Alur Eksperimen

Alur dari keseluruhan eksperimen yang dilakukan untuk memperoleh hasil dapat dilihat pada Gambar 2 berikut.



Gambar 2 Alur Eksperimen

Pada Gambar 2, pengujian dimulai dengan menyiapkan perangkat keras sesuai ruang lingkup penelitian, mengonfigurasi aplikasi backend untuk 12 kombinasi bahasa pemrograman dan metode *query*, serta menyusun skrip pengujian untuk mengontrol *mouse* dan *keyboard* agar simulasi berjalan presisi. Eksperimen diawali dengan menjalankan *script* pengujian, dimulai dengan monitoring CPU dan memori menggunakan Glances, diikuti jeda 10 detik sebelum aplikasi backend dijalankan. Setelah jeda tambahan 10 detik, simulasi pengujian dilakukan menggunakan K6. Usai simulasi, sistem distabilkan dengan jeda 20 detik sebelum aplikasi backend dihentikan, lalu monitoring CPU dan memori dihentikan setelah jeda 10 detik.

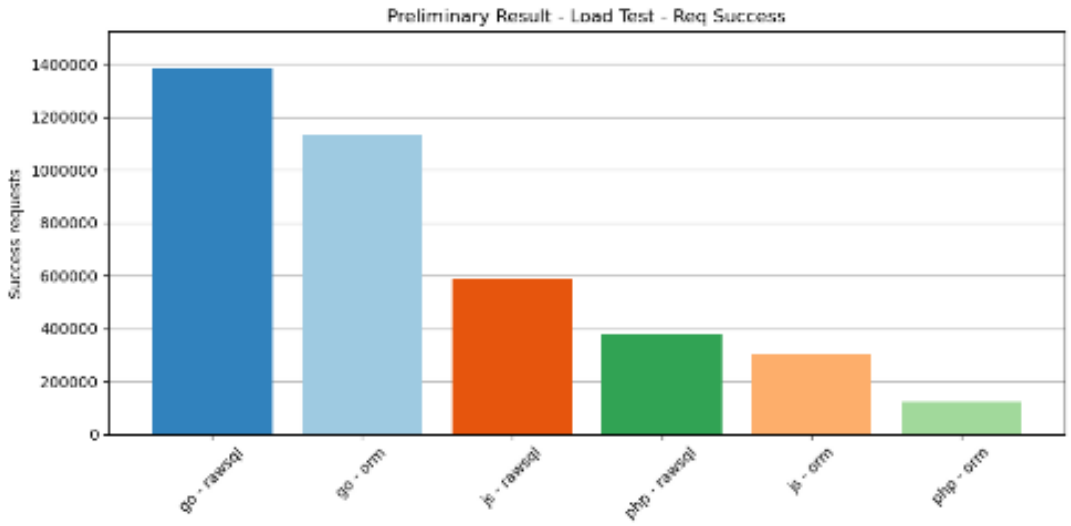
Hasil pengujian disimpan dalam format .txt dan .csv, kemudian perangkat keras di-*reboot* untuk memastikan lingkungan bersih dari *cache* dan faktor internal lainnya sebelum pengujian berikutnya dilakukan. Proses ini diulang hingga semua kombinasi backend dan simulasi selesai. Terakhir, hasil dianalisis dengan memproyeksikan data untuk menarik kesimpulan dari eksperimen.

3. HASIL DAN PEMBAHASAN

3.1 Eksperimen Sederhana

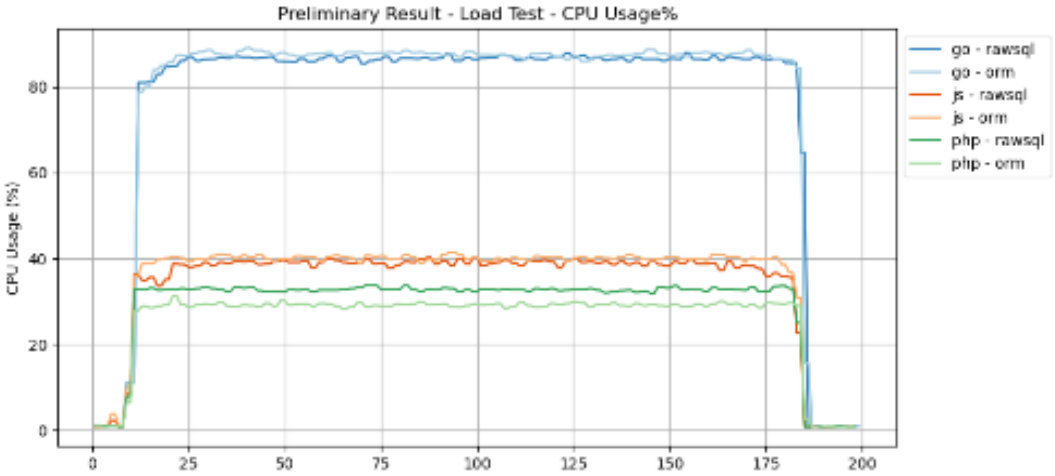
Sebagai langkah awal dalam penelitian ini, dilakukan eksperimen sederhana untuk memperoleh hipotesis awal mengenai pengaruh kombinasi bahasa pemrograman dan metode *query* terhadap kinerja backend. Eksperimen ini memiliki ruang lingkup yang mirip dengan eksperimen utama, namun dengan skala yang lebih kecil, yaitu menggunakan satu tabel *users* dan hanya menerapkan metode *query* Raw SQL serta ORM.

Pendekatan ini bertujuan untuk memahami bagaimana setiap kombinasi memengaruhi efisiensi pemrosesan data sebelum eksperimen utama yang lebih kompleks dilakukan. Dengan demikian, penelitian dapat mengidentifikasi pola awal perbedaan kinerja dari berbagai kombinasi bahasa pemrograman dan metode *query*, sehingga dapat merumuskan hipotesis awal penelitian. Hasil dari eksperimen sederhana ini diharapkan menjadi dasar dalam merancang pengujian yang lebih komprehensif. Rincian hasil eksperimen sederhana dapat dilihat pada Gambar 3 hingga 5.



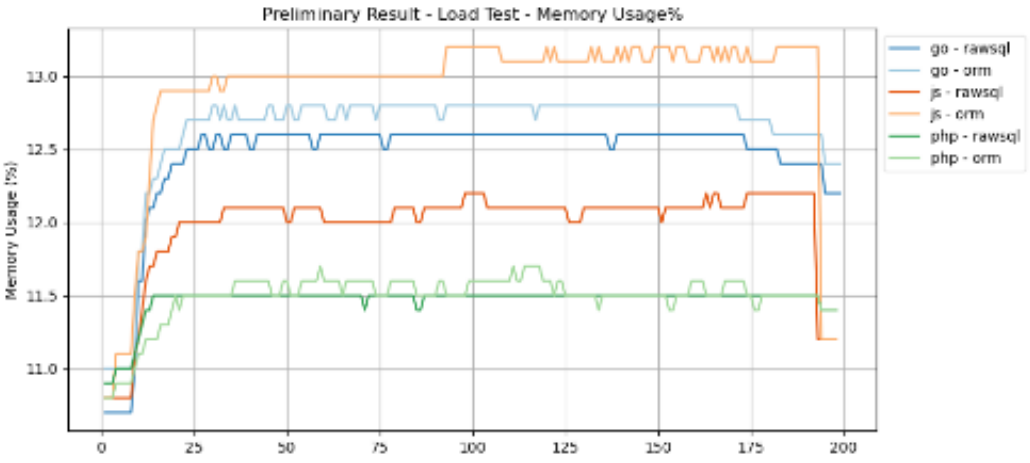
Gambar 3 Hasil Eksperimen Sederhana: Success Requests Total

Pada Gambar 3, hasil eksperimen sederhana untuk total permintaan yang berhasil menunjukkan bahwa Go dengan Raw SQL memiliki kinerja tertinggi. Metode *query* Raw SQL juga unggul dibandingkan ORM di semua bahasa pemrograman yang diuji.



Gambar 4 Hasil Eksperimen Sederhana: CPU Usage

Pada Gambar 4, hasil eksperimen sederhana untuk penggunaan CPU menunjukkan bahwa bahasa Go memiliki konsumsi CPU tertinggi dibandingkan JavaScript dan PHP.



Gambar 5 Hasil Eksperimen Sederhana: Memory Usage

Pada Gambar 5, hasil eksperimen sederhana untuk penggunaan memori menunjukkan bahwa bahasa JavaScript dengan metode *query* ORM memiliki konsumsi memori tertinggi, diikuti oleh Go dengan kedua metode *query*, JavaScript dengan Raw SQL, serta PHP dengan kedua metode *query* yang memiliki konsumsi memori terendah.

3.2 Perumusan Hipotesis

Berdasarkan eksperimen sederhana yang telah dilakukan, dirumuskan beberapa hipotesis sebagai dasar penelitian ini, yaitu sebagai berikut:

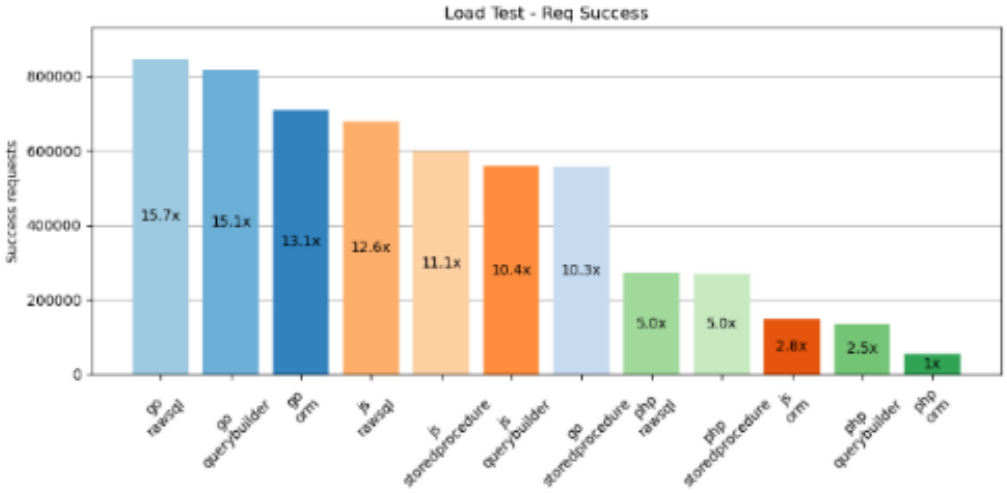
Hipotesis 1 (H_1): Kinerja bahasa pemrograman Go dengan metode *query* Raw SQL menghasilkan jumlah permintaan berhasil yang lebih tinggi dibandingkan kombinasi bahasa pemrograman dan metode *query* lainnya.

Hipotesis 2 (H_2): Kinerja metode *query* Raw SQL menghasilkan jumlah permintaan berhasil yang lebih tinggi pada ketiga bahasa pemrograman yang diuji dibandingkan metode *query* lainnya.

Hipotesis 3 (H_3): Kinerja Bahasa pemrograman PHP memiliki penggunaan CPU dan penggunaan memori yang paling rendah dibandingkan dengan bahasa pemrograman lainnya.

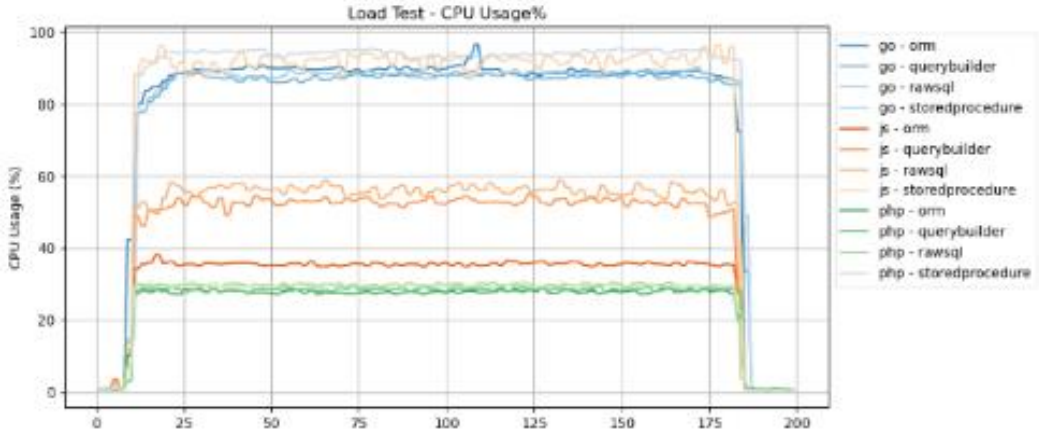
3.4 Eksperimen Utama

Setelah memperoleh hipotesis awal dari eksperimen sederhana, dilakukan eksperimen utama untuk menguji pengaruh kombinasi bahasa pemrograman dan metode *query* terhadap kinerja backend secara lebih mendalam. Eksperimen ini mencakup skenario yang lebih kompleks agar hasilnya lebih akurat dan representatif dalam berbagai kondisi. Hasil dari eksperimen utama yang telah dilakukan ditunjukkan pada Gambar 6 hingga 14.



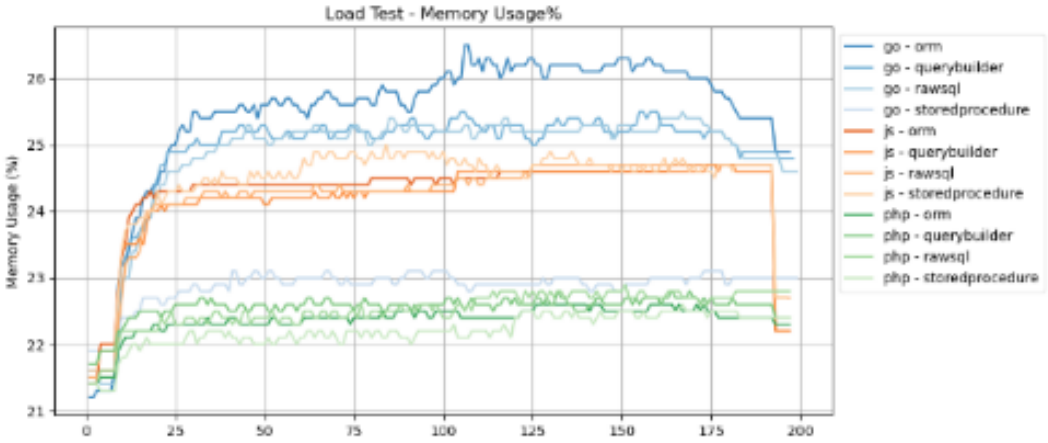
Gambar 6 Hasil Load Testing: Success Requests Total

Pada Gambar 6, hasil simulasi Load Testing menunjukan bahwa kombinasi bahasa pemrograman Go dengan metode *query* Raw SQL memiliki kinerja yang secara signifikan lebih tinggi dibandingkan kombinasi lainnya. Selain itu, Go cenderung unggul dalam hal kinerja jika dibandingkan dengan JavaScript dan PHP. Lalu untuk metode *query* Raw SQL, terbukti lebih cepat dibandingkan metode *query* lainnya pada ketiga bahasa pemrograman tersebut.



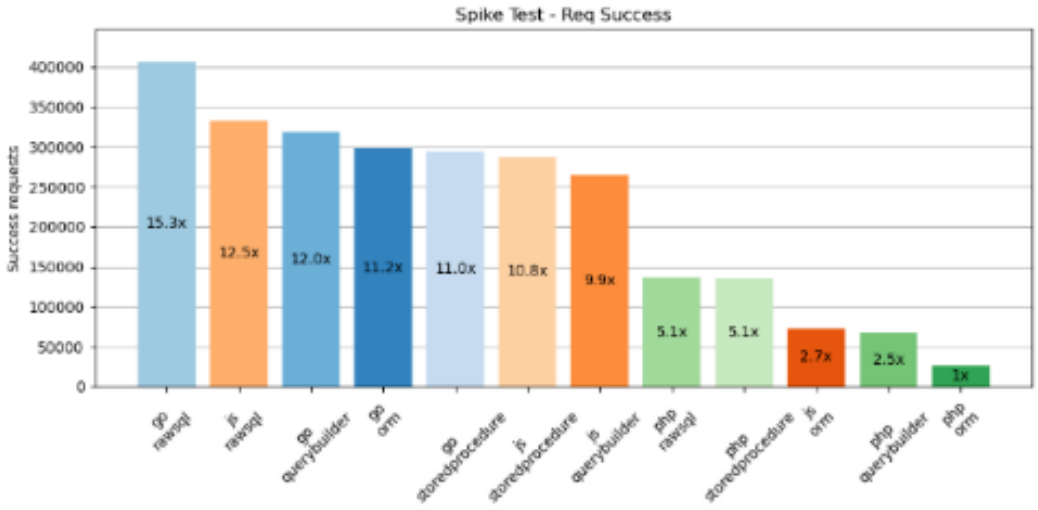
Gambar 7 Hasil Load Testing: CPU Usage

Pada Gambar 7, hasil pengukuran penggunaan CPU pada simulasi Load Testing menunjukkan pola penggunaan sumber daya yang berbeda pada masing-masing kombinasi bahasa pemrograman dan metode *query*. Secara khusus, bahasa pemrograman Go dengan semua metode *query*, serta JavaScript dengan metode *query* Stored Procedure, memperlihatkan tingkat penggunaan CPU yang relatif tinggi. Sebaliknya, PHP dengan seluruh metode *query* yang diterapkan tercatat memiliki tingkat penggunaan CPU yang cenderung lebih rendah.



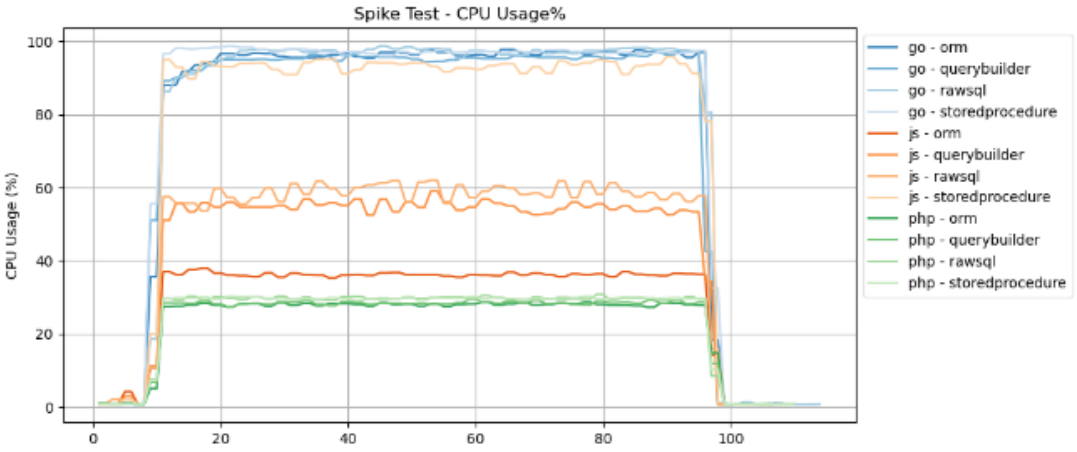
Gambar 8 Hasil Load Testing: Memory Usage

Pada Gambar 8, hasil pengukuran penggunaan memori pada simulasi Load Testing menunjukkan variasi yang signifikan antar kombinasi bahasa pemrograman dan metode *query*. Secara umum, Go dengan seluruh metode *query* selain Stored Procedure memperlihatkan tingkat penggunaan memori yang relatif tinggi. Sementara itu, PHP dengan semua metode *query* dan Go dengan metode *query* Stored Procedure cenderung memiliki penggunaan memori yang lebih rendah dibandingkan kombinasi bahasa pemrograman dan metode *query* lainnya.



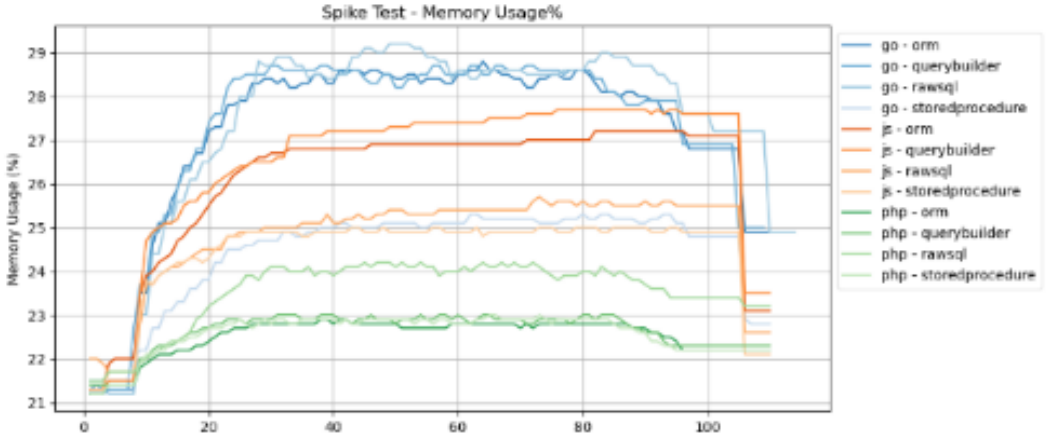
Gambar 9 Hasil Spike Testing: Success Requests Total

Pada Gambar 9, hasil simulasi Spike Testing menunjukkan bahwa Go dengan metode *query* Raw SQL masih mendominasi kinerja dalam menangani lonjakan beban secara tiba-tiba. Di bawahnya, JavaScript dengan metode Raw SQL mencatat kinerja yang relatif kuat, disusul oleh kombinasi Go dengan metode *query* lainnya. Sementara itu, PHP dengan *query* ORM kembali berada di posisi terendah, menandakan keterbatasan dalam merespons peningkatan beban mendadak.



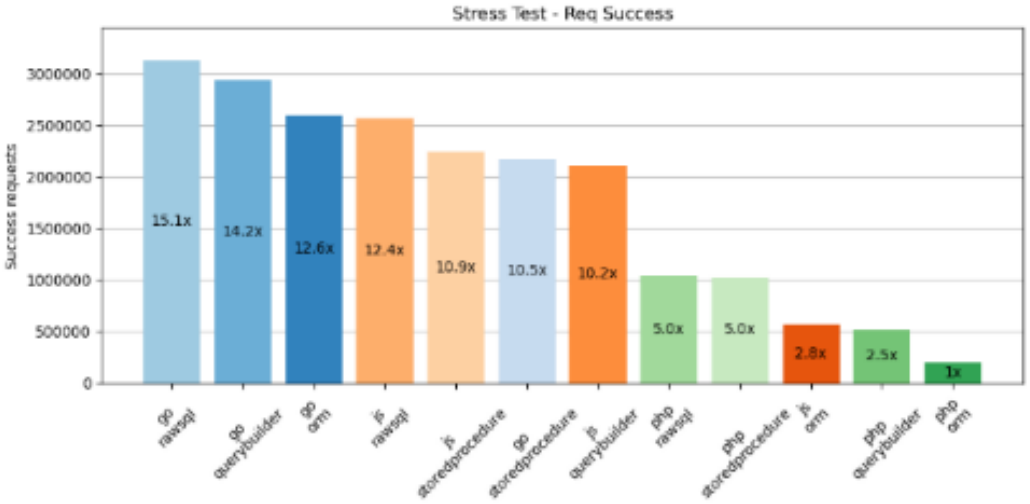
Gambar 10 Hasil Spike Testing: CPU Usage

Pada Gambar 10, hasil pengujian penggunaan CPU pada simulasi Spike Testing menunjukkan bahwa seluruh kombinasi dengan bahasa Go, serta JavaScript dengan metode *query* Stored Procedure, cenderung berada pada tingkat penggunaan CPU yang relatif tinggi. Di sisi lain, PHP dengan semua metode *query* yang diuji terlihat cenderung rendah dalam memanfaatkan sumber daya CPU. Sementara itu, JavaScript dengan metode ORM juga mencatat tingkat penggunaan CPU yang rendah, hampir menyamai tingkat penggunaan CPU bahasa pemrograman PHP.



Gambar 11 Hasil Spike Testing: Memory Usage

Pada Gambar 11, hasil pengukuran penggunaan memori pada simulasi Spike Testing menunjukkan pola yang hampir sejalan dengan temuan pada simulasi Load Testing sebelumnya. Seluruh kombinasi bahasa pemrograman Go, kecuali yang menggunakan metode *query* Stored Procedure, cenderung memiliki tingkat penggunaan memori yang relatif tinggi. Sebaliknya, seluruh kombinasi dengan bahasa PHP konsisten menunjukkan penggunaan memori yang rendah.



Gambar 12 Hasil Stress Testing: Success Requests Total

Pada Gambar 12, hasil simulasi Stress Testing memperlihatkan pola yang mirip dengan hasil pada simulasi Load Testing. Go dengan seluruh metode *query*, kecuali Stored Procedure, kembali menunjukkan kinerja yang paling tinggi. Sementara itu, PHP dengan metode ORM tetap menempati posisi paling rendah dalam pengujian ini.

Gambar 13 Hasil Stress Testing: CPU Usage

Pada Gambar 13, hasil pengujian penggunaan CPU pada simulasi Stress Testing memperlihatkan pola yang serupa dengan simulasi Spike Testing sebelumnya. Seluruh kombinasi Go serta JavaScript dengan metode Stored Procedure cenderung menunjukkan kinerja yang tinggi. Sementara itu, PHP pada semua metode *query* kembali menunjukkan kinerja yang cenderung rendah. JavaScript dengan metode ORM juga memiliki kinerja yang relatif rendah, hampir menyamai tingkat kinerja PHP.

Gambar 14 Hasil Stress Testing: Memory Usage

Pada Gambar 14, hasil pengukuran penggunaan memori pada simulasi Stress Test menunjukkan pola yang serupa dengan simulasi Load Testing sebelumnya. Go, pada seluruh metode *query* kecuali Stored Procedure, kembali menampilkan tingkat konsumsi memori yang tinggi. JavaScript terlihat berada di tingkat penggunaan memori yang menengah, sedangkan PHP secara konsisten menunjukkan konsumsi memori terendah di antara seluruh metode *query* yang diuji.

4. KESIMPULAN DAN SARAN

Berdasarkan hasil pengujian yang dilakukan, terlihat pola yang konsisten antara berbagai skenario pengujian (Load Testing, Spike Testing, dan Stress Testing) terhadap kombinasi bahasa pemrograman dan metode *query*. Go dengan metode Raw SQL menunjukkan kinerja tertinggi dalam hal respons terhadap beban, penanganan lonjakan mendadak, serta efisiensi waktu eksekusi. Sebaliknya, PHP dengan metode ORM menunjukkan kinerja paling rendah, mengindikasikan adanya hambatan akibat tingginya abstraksi ORM. JavaScript berada di posisi menengah, dengan kinerja yang lebih baik dari PHP namun masih di bawah Go, terutama saat menggunakan ORM. Dari hasil tersebut, hipotesis 1 diterima.

Dari segi penggunaan sumber daya, Go dan JavaScript (dengan Stored Procedure) menunjukkan pemanfaatan CPU yang tinggi, menandakan proses yang berat namun efektif dalam menangani beban besar. Sementara itu, PHP dengan semua metode *query* cenderung rendah dalam pemanfaatan CPU dan memori, yang dapat diartikan sebagai efisiensi atau keterbatasan dalam menangani beban tinggi. Dari hasil tersebut, hipotesis 2 dan 3 diterima.

Untuk penelitian selanjutnya, disarankan untuk menguji bahasa pemrograman lain seperti Python dan Rust guna mengeksplorasi kinerja backend lebih lanjut. Selain itu, pengujian dapat difokuskan pada penerapan pola desain seperti DAO atau prinsip SOLID untuk melihat dampaknya terhadap efisiensi sistem. Soak Testing juga perlu dipertimbangkan guna mengidentifikasi potensi masalah kestabilan aplikasi dalam jangka panjang.

DAFTAR PUSTAKA

[1] A. Godinho, J. Rosado, F. Sa, and F. Cardoso, “Performance Comparison of RESTful Web APIs using a Test Suite: .NET vs. Java Spring Boot,” *Journal of Software and Systems Development*, pp. 1–32, Aug. 2024, doi: 10.5171/2024.478010.

[2] A. Ehsan, M. A. M. E. Abuhaliqa, C. Catal, and D. Mishra, “RESTful API Testing Methodologies: Rationale, Challenges, and Solution Directions,” May 01, 2022, *MDPI*. doi: 10.3390/app12094369.

[3] M. Iqbal and Nurwati, "PENERAPAN SISTEM TERINTEGRASI MENGGUNAKAN RESTFUL API PADA DEALER MANAGEMENT SYSTEM PANCA NIAGA SEI PIRING," Feb. 2023. [Online]. Available: http://jurnal.goretanpena.com/index.php/JSSR

[4] A. Niarman, Iswandi, and A. K. Candri, "Comparative Analysis of PHP Frameworks for Development of Academic Information System Using Load and Stress Testing," *International Journal Software Engineering and Computer Science (IJSECS)*, vol. 3, no. 3, pp. 424–436, Dec. 2023, doi: 10.35870/ijsecs.v3i3.1850.

[5] F. Pamungkas and H. Setiaji, "IMPLEMENTASI CLEAN ARCHITECTURE PADA PEMBUATAN API MENGGUNAKAN GOLANG," Apr. 2024.

[6] B. Basumatary and N. Agnihotri, "Benefits and Challenges of Using NodeJS," *International Journal of Innovative Research in Computer Science & Technology*, pp. 67–70, May 2022, doi: 10.55524/ijircst.2022.10.3.13.

[7] H. Brar, T. Kaur, and Y. Rajoria, "The Better Comparison between PHP, Python-web & Node.js," Jun. 2021. [Online]. Available: www.ijres.org

[8] A. Ismail, A. Y. Ananta, S. N. Arief, and E. N. Hamdana, "Performance Testing Sistem Ujian Online Menggunakan Jmeter Pada Lingkungan Virtual," Feb. 2023.

[9] W. Tejaya, S. Rahman, and A. Munir, "PENGUJIAN WEBSITE INVITEES MENGGUNAKAN METODE LOAD TESTING DENGAN APACHE JMETER," Aug. 2023, [Online]. Available: https://jurnal.kharisma.ac.id/kharismatech/

[10] M. Hendayun, A. Ginanjar, and Y. Ihsan, "ANALYSIS OF APPLICATION PERFORMANCE TESTING USING LOAD TESTING AND STRESS TESTING METHODS IN API SERVICE," *JURNAL SISFOTEK GLOBAL*, vol. 13, no. 1, p. 28, Mar. 2023, doi: 10.38101/sisfotek.v13i1.2656.

[11] M. R. Azzaky, A. Pinandito, and M. A. Akbar, "Analisis Penggunaan Raw Query dan ORM pada Aplikasi API Berbasis Bahasa Pemrograman Go," Jun. 2024. [Online]. Available: http://j-ptiik.ub.ac.id

[12] A. D. Septiadi and L. Jeong-Bae, "Comparative Analysis of Database Query Storage Performance Between Stored Procedure and Function," *International Journal of Informatics and Information System*, vol. 3, no. 2, pp. 60–66, 2020.

[13] D. I. Permatasari *et al.*, "PENGUKURAN THROUGHPUT LOAD TESTING MENGGUNAKAN TEST CASE SAMPLING GORILLA TESTING," *Seminar Nasional Sistem Informasi*, no. 4, 2019.

[14] H. Ardiansyah and A. Fatwanto, "Comparison of Memory usage between REST API in Javascript and Golang," *MATRIK : Jurnal Manajemen, Teknik Informatika dan Rekayasa Komputer*, vol. 22, no. 1, pp. 229–240, Nov. 2022, doi: 10.30812/matrik.v22i1.1325.