JURNAL KOMPUTER TEKNOLOGI INFORMASI SISTEM KOMPUTER

p-ISSN: 2962-3022 | e-ISSN: 2963-7104
Volume 4 No 3 Februari 2022
Hal: 1583-1590
http://ejurnal.lkpkaryaprima.id/index.php/juktisi
DOI: https://doi.org/10.62712/juktisi.v4i3.717

Analisis Komparatif Kinerja Laravel Octane dan Webman pada Layanan API Berbasis Worker Model PHP

Ade Iskandar[1,*], Muhammad Irfan Sarif[2], Muhammad Fuad Hafiz[3], Dedy Rahman Harahap[4] , Samuel Sampe Tuah Purba[5]

[1,2,3,4,5] Magister Teknologi Informasi, Universitas Pembangunan Panca Budi, Medan, Indonesia
Email: [1,*]suratiskandar@gmail.com, [2]irfanberbagi@gmail.com, [3]fizgilang@gmail.com, [4]ekyr422@gmail.com, [5]samuelpurba029@gmail.com
(* Email Corresponding Author: suratiskandar@gmail.com)
Received: 8 Desember 2025 | Revision: 12 Desember 2025 | Accepted: 12 Desember 2025

Abstrak

Performa server API merupakan faktor krusial dalam pengembangan aplikasi web modern yang menuntut latensi rendah dan throughput tinggi. Penelitian ini bertujuan membandingkan kinerja Laravel Octane berbasis FrankenPHP dan Webman berbasis Workerman sebagai framework PHP worker model pada operasi READ dan CREATE data. Metode penelitian menggunakan pendekatan kuantitatif eksperimental melalui *benchmarking* dengan ApacheBench pada dua skenario API, yaitu pengambilan dan penyimpanan 1000 baris data MySQL, dengan konfigurasi tingkat konkurensi terstandar pada lingkungan server identik. Parameter kinerja yang dievaluasi meliputi *requests per second*, median latency, dan P95 latency. Hasil penelitian menunjukkan bahwa Webman secara konsisten unggul pada kedua skenario pengujian. Pada operasi READ, Webman mencapai throughput hampir dua kali lipat dibanding Laravel Octane dengan latensi median lebih rendah hingga 65%. Sementara itu, pada operasi CREATE, Webman mencatat throughput 1,7 kali lebih tinggi dan waktu respons sekitar 50% lebih cepat. Keunggulan ini berkaitan erat dengan arsitektur event-driven non-blocking Workerman yang meminimalkan overhead eksekusi. Laravel Octane tetap menampilkan stabilitas kinerja yang baik, namun kompleksitas internal framework menyebabkan latensi yang relatif lebih tinggi. Penelitian ini menyimpulkan bahwa Webman lebih sesuai untuk pengembangan API bertrafik tinggi dan sistem mikroservis, sedangkan Laravel Octane unggul untuk aplikasi kompleks berbasis ekosistem Laravel. Temuan ini memberikan kontribusi empiris bagi pemilihan framework PHP berdasarkan kebutuhan performa aktual.

Kata Kunci: *Laravel Octane, Webman, Workerman, FrankenPHP, Benchmark API PHP.*

Abstract

Server API performance is a crucial factor in the development of modern web applications that demand low latency and high throughput. This research aims to compare the performance of Laravel Octane based on FrankenPHP and Webman based on Workerman as PHP worker model frameworks for READ and CREATE data operations. The research method uses a quantitative experimental approach thru benchmarking with ApacheBench on two API scenarios: retrieving and storing 1000 rows of MySQL data, with standardized concurrency level configurations in identical server environments. The performance parameters evaluated include requests per second, median latency, and P95 latency. The research results show that Webman consistently outperformed in both testing scenarios. In READ operations, Webman achieved nearly double the throughput compared to Laravel Octane, with a median latency up to 65% lower. Meanwhile, in CREATE operations, Webman recorded 1.7 times higher throughput and response times about 50% faster. This advantage is closely related to Workerman's non-blocking event-driven architecture, which minimizes execution overhead. Laravel Octane continues to demonstrate good performance stability, but the framework's internal complexity leads to relatively higher latency. This research concludes that Webman is more suitable for high-traffic API development and microservices systems, while Laravel Octane excels for complex applications built on the Laravel ecosystem. This finding provides empirical contributions for selecting a PHP framework based on actual performance needs..

Keywords*: Laravel Octane, Webman, Workerman, FrankenPHP, Benchmark API PHP.*

1. PENDAHULUAN

Perkembangan teknologi informasi yang pesat telah mendorong transformasi signifikan dalam pengembangan aplikasi web modern, khususnya pada lapisan backend yang bertugas memproses permintaan klien dalam jumlah besar dengan respons yang cepat dan andal. API berbasis web menjadi fondasi utama integrasi layanan digital, mulai dari sistem e-commerce, aplikasi keuangan, layanan pendidikan daring, hingga platform Internet of Things (IoT)[1], [2]. Dalam konteks ini, performa backend bukan sekadar persoalan kecepatan, melainkan juga menyangkut stabilitas sistem, pengalaman pengguna, dan efisiensi pemanfaatan sumber daya server[3]. Meningkatnya volume trafik dan tuntutan real-time processing menempatkan kinerja server API sebagai faktor kunci keberhasilan aplikasi berskala menengah hingga besar[4]. Berbagai studi terkini menegaskan bahwa latensi yang rendah dan throughput yang tinggi berkorelasi langsung dengan kepuasan pengguna dan tingkat adopsi layanan digital. Oleh sebab itu, optimasi arsitektur backend menjadi bidang penelitian yang sangat relevan, khususnya pada platform PHP yang masih mendominasi ekosistem pengembangan web global karena fleksibilitasnya, kematangan komunitas, dan dukungan framework yang luas[5].

JURNAL KOMPUTER TEKNOLOGI INFORMASI SISTEM KOMPUTER
p-ISSN: 2962-3022 | e-ISSN: 2963-7104
Volume 4 No 3 Februari 2022
Hal: 1583-1590
http://ejurnal.lkpkaryaprima.id/index.php/juktisi
DOI: https://doi.org/10.62712/juktisi.v4i3.717

Meskipun PHP telah berkembang signifikan secara performa melalui pembaruan mesin Zend Engine dan dukungan opcode caching, arsitektur tradisional berbasis PHP-FPM masih membawa keterbatasan inheren. Setiap permintaan HTTP diproses melalui siklus bootstrap aplikasi penuh, menciptakan overhead eksekusi yang relatif besar, terutama saat menangani beban masif dan request berulang. Literatur terkini menunjukkan bahwa model request response stateless ini tidak efisien untuk skenario high concurrency dan beban API intensif, seperti pengambilan data masif maupun operasi transaksi database berulang. Untuk menjawab tantangan ini, pendekatan worker model diperkenalkan dalam ekosistem PHP melalui berbagai solusi seperti Swoole, RoadRunner, Workerman, hingga FrankenPHP. Konsep worker model mengandalkan proses aplikasi yang persisten di memori, memungkinkan reuse konteks aplikasi tanpa perlu memuat ulang komponen inti pada setiap request. Pendekatan ini terbukti dalam berbagai studi mampu meningkatkan throughput secara signifikan sekaligus menurunkan latensi, terutama pada layanan API yang menuntut performa tinggi. Namun, implementasi dan efektivitas worker model sangat bergantung pada framework dan mesin runtime yang digunakan, sehingga membuka ruang kajian komparatif yang mendalam.

Dua solusi yang memperoleh perhatian besar dalam komunitas pengembang PHP saat ini adalah **Laravel Octane dengan FrankenPHP** dan **Webman dengan Workerman**. Laravel Octane merupakan ekstensi resmi ekosistem Laravel yang memanfaatkan worker model untuk mempertahankan instans aplikasi tetap aktif di memori. Dengan dukungan runtime seperti Swoole dan FrankenPHP, Octane menawarkan peningkatan performa signifikan sembari mempertahankan kenyamanan, struktur MVC, serta kelengkapan fitur Laravel seperti middleware, dependency injection container, ORM Eloquent, dan sistem routing yang matang. Di sisi lain, Webman dikembangkan sebagai framework PHP ringan berbasis Workerman yang mengutamakan performa tinggi menggunakan arsitektur event-driven non-blocking I/O. Webman mengadopsi desain minimalis dengan penekanan pada efisiensi eksekusi dan pengurangan overhead lapisan framework. Walaupun kedua solusi mengklaim keunggulan performa berbasis worker model, terdapat perbedaan fundamental pada filosofi desain: Laravel Octane memprioritaskan kemudahan pengembangan dan kestabilan ekosistem, sedangkan Webman berfokus pada efisiensi eksekusi dan throughput maksimal.

Permasalahan utama penelitian ini terletak pada masih minimnya kajian ilmiah komparatif yang secara spesifik membandingkan kinerja kedua framework tersebut dalam konteks beban API yang realistis, terukur, dan terstandarisasi[6]. Sebagian besar referensi yang tersedia bersifat dokumentasi resmi, blog teknis, atau laporan informal komunitas, yang seringkali tidak melalui metode benchmarking terkontrol[7]. Akibatnya, klaim performa yang beredar belum memiliki validitas akademik yang kuat dan sulit dijadikan rujukan objektif dalam perencanaan sistem backend berskala besar. Selain itu, banyak studi sebelumnya menguji performa hanya pada operasi sederhana seperti endpoint statis atau request tanpa interaksi database intensif, sehingga kurang merepresentasikan kondisi produksi nyata yang sarat dengan eksekusi query kompleks, serialisasi data, serta beban I/O simultan. Dengan demikian, masih terdapat celah penelitian yang signifikan dalam hal analisis performa framework PHP worker model pada beban CRUD API yang realistis[8].

Sebagai solusi umum terhadap permasalahan ini, pendekatan benchmarking berbasis simulasi beban (load testing) digunakan secara luas dalam literatur teknik perangkat lunak untuk menilai kinerja sistem aplikasi web[9]. Metode ini melibatkan pemanfaatan alat uji performa untuk mengirimkan permintaan HTTP dalam jumlah besar terhadap endpoint tertentu dan mengamati metrik kunci seperti requests per second (RPS), latency median, percentile latency (P95 dan P99), tingkat kegagalan permintaan, dan transfer rate data. Benchmarking memungkinkan perbandingan objektif antar framework dengan kondisi lingkungan uji yang identik, sehingga variabel yang memengaruhi performa dapat diminimalkan[10]. Berbagai penelitian sebelumnya telah menerapkan metode ini untuk membandingkan performa Node.js, Go, Python FastAPI, dan framework PHP konvensional berbasis PHP-FPM. Hasilnya secara konsisten menunjukkan keunggulan model event-loop dan worker persistence dibanding pendekatan tradisional stateless. Namun demikian, sangat sedikit studi yang menerapkan metode benchmarking komprehensif untuk membandingkan Laravel Octane dan Webman secara langsung dalam konteks operasi database masif.

Dalam literatur yang ada, Laravel Octane dinilai berhasil meningkatkan performa aplikasi Laravel hingga beberapa kali lipat dibanding PHP-FPM dengan mempertahankan kenyamanan developer experience. Studi teknis juga menyoroti bahwa FrankenPHP, sebagai runtime modern berbasis Caddy dan PHP embed, mampu menawarkan stabilitas memori serta integrasi native HTTP server yang lebih efisien. Akan tetapi, beberapa laporan komunitas mengindikasikan bahwa kompleksitas arsitektur Laravel, terutama pada dependensi container, middleware stack, dan ORM, tetap menimbulkan overhead non-trivial pada worker model[11]. Sebaliknya, Webman dan Workerman dilaporkan memiliki keunggulan struktural melalui event loop yang dieksekusi mendekati level C, pemrosesan non-blocking, serta desain framework yang minimalis tanpa lapisan middleware berat. Beberapa publikasi teknis mencatat RPS yang tinggi pada Webman dalam skenario high concurrency, meskipun trade-off dalam ekosistem dan kemudahan pengembangan sering menjadi catatan kritis. Namun, sebagian besar temuan tersebut belum dirumuskan secara akademik melalui pengujian terkontrol pada operasi database CRUD berskala besar[12].

Kajian literatur yang mengaitkan langsung perbandingan antara Laravel Octane dan Webman masih bersifat terbatas, fragmentaris, dan tidak konsisten dalam metodologi pengujian. Mayoritas referensi hanya menilai throughput permintaan statis atau endpoint ringan tanpa beban database signifikan. Belum ditemukan penelitian komprehensif yang menguji performa kedua framework pada dua operasi inti API modern, yakni **READ** (mengambil sejumlah besar data) dan **CREATE** (menyimpan data masif), dengan konfigurasi server identik dan parameter benchmarking terstandarisasi. Kekosongan penelitian ini menjadi semakin relevan karena operasi CRUD merupakan inti aktivitas layanan API produksi,

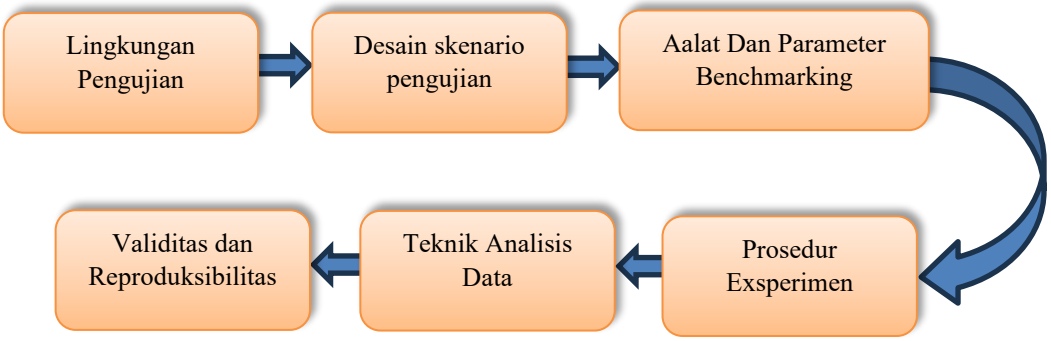
JURNAL KOMPUTER TEKNOLOGI INFORMASI SISTEM KOMPUTER
p-ISSN: 2962-3022 | e-ISSN: 2963-7104
Volume 4 No 3 Februari 2022
Hal: 1583-1590
http://ejurnal.lkpkaryaprima.id/index.php/juktisi
DOI: https://doi.org/10.62712/juktisi.v4i3.717

menentukan kestabilan sistem dan pengalaman pengguna akhir. Dengan demikian, terdapat kesenjangan penelitian jelas terkait belum adanya evaluasi empiris yang memetakan secara objektif efektivitas Laravel Octane dan Webman untuk skenario API intensif berbasis worker model[13].

Berdasarkan celah penelitian tersebut, tujuan studi ini adalah melakukan analisis komparatif kinerja antara Laravel Octane (menggunakan runtime FrankenPHP) dan Webman (menggunakan Workerman) dalam menjalankan operasi READ dan CREATE pada API berbasis PHP worker model terhadap dataset berukuran besar. Kebaruan penelitian terletak pada penerapan benchmarking terkontrol menggunakan beban simultan terukur pada operasi database realistis, bukan sekadar endpoint statis, sehingga menghasilkan data empiris yang relevan dengan kondisi produksi nyata. Studi ini menguji hipotesis bahwa arsitektur event-loop ringan Workerman memberikan keunggulan performa signifikan dalam throughput dan latensi dibandingkan Laravel Octane, sementara Laravel tetap unggul dari sisi kestabilan dan kemudahan pengembangan[14]. Ruang lingkup penelitian dibatasi pada pengujian dua operasi inti CRUD (READ dan CREATE) menggunakan lingkungan server dan inspeksi metrik performa standar, dengan tujuan menghasilkan rekomendasi rasional bagi pengembang dan perancang sistem dalam memilih framework PHP worker model yang sesuai dengan kebutuhan aplikasi API berskala besar[15].

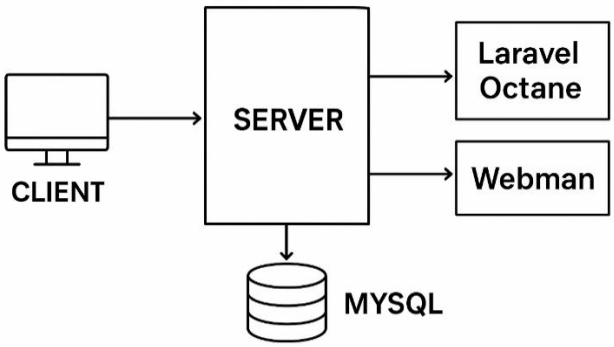
2. METODOLOGI PENELITIAN

Penelitian ini menggunakan pendekatan kuantitatif eksperimental dengan metode *benchmarking performance testing* untuk membandingkan kinerja dua framework PHP berbasis *worker model*, yaitu Laravel Octane (menggunakan runtime FrankenPHP) dan Webman (menggunakan engine Workerman). Pendekatan ini dipilih karena mampu menghasilkan data empiris yang objektif, terukur, dan dapat direplikasi, sehingga sesuai untuk mengevaluasi performa sistem backend dalam kondisi terkontrol. Eksperimen dirancang untuk menguji performa pada dua skenario utama operasi API, yakni operasi **READ** untuk mengambil data dalam jumlah besar dan operasi **CREATE** untuk melakukan penyimpanan data secara masif, yang merupakan representasi aktivitas inti layanan API pada aplikasi produksi.



Gambar 1. Struktur Penelitian

2.1 Lingkungan Pengujian



Gambar 2. Diagram struktur lingkungan pengujian API PHP worker model

Pengujian dilakukan pada satu unit perangkat keras yang sama untuk memastikan konsistensi kondisi eksperimen dan meminimalkan variabilitas hasil yang disebabkan oleh perbedaan spesifikasi sistem. Perangkat uji yang digunakan adalah laptop dengan prosesor Apple Silicon M1, arsitektur ARM64, dan sistem operasi macOS. Platform perangkat lunak terdiri atas PHP versi 8.2 sebagai bahasa pemrograman utama, MySQL 8 sebagai sistem manajemen basis data relasional, serta masing-masing framework PHP yang diuji, yaitu Laravel versi 12 yang dikombinasikan dengan

JURNAL KOMPUTER TEKNOLOGI INFORMASI SISTEM KOMPUTER
p-ISSN: 2962-3022 | e-ISSN: 2963-7104
Volume 4 No 3 Februari 2022
Hal: 1583-1590
http://ejurnal.lkpkaryaprima.id/index.php/juktisi
DOI: https://doi.org/10.62712/juktisi.v4i3.717

Laravel Octane dan runtime FrankenPHP, serta Webman versi 1.x yang berjalan di atas engine Workerman versi 5.x. Seluruh dependensi dikelola melalui Composer dengan konfigurasi *default production mode* tanpa fitur debug untuk menghindari beban tambahan yang dapat memengaruhi hasil benchmark.

Server HTTP pada Laravel Octane dikendalikan oleh FrankensteinPHP yang terintegrasi dengan web server Caddy sebagai layer HTTP native, sementara Webman menggunakan sistem socket server bawaan Workerman dengan arsitektur *event-driven non-blocking I/O*. Konfigurasi worker pada kedua framework ditetapkan pada nilai bawaan yang direkomendasikan dokumentasi resmi masing-masing, tanpa modifikasi berlebihan, guna mencerminkan kondisi penerapan standar yang realistis di lingkungan pengembangan maupun produksi awal.

2.2 Desain Skenario Pengujian

Desain pengujian difokuskan pada dua skenario utama berdasarkan aktivitas API yang umum digunakan di sistem layanan data:

a. Operasi READ

Endpoint GET dirancang untuk melakukan pengambilan data sebanyak 1000 baris dari tabel MySQL menggunakan query SELECT. Data hasil query kemudian diserialisasi dalam format JSON sebagai respons API terhadap klien. Skenario ini mewakili aktivitas data retrieval intensif yang lazim terjadi pada aplikasi e-commerce, dashboard analitik, atau sistem manajemen konten.

b. Operasi CREATE

Endpoint POST dirancang untuk menambahkan 1000 baris data baru ke dalam database menggunakan perintah INSERT. Setiap permintaan melakukan batch insert data yang telah dipersiapkan secara otomatis dari sisi server untuk menjaga konsistensi format dan validasi input. Operasi ini merepresentasikan aktivitas transaksi atau input data masif seperti pencatatan log, pendaftaran pengguna, atau sinkronisasi data eksternal.

Seluruh endpoint API pada kedua framework dibuat dengan struktur logika identik, baik dari sisi pengelolaan request, validasi data, eksekusi query, hingga pembentukan respons JSON. Kesamaan desain endpoint bertujuan untuk memastikan bahwa perbedaan performa yang terukur murni berasal dari perbedaan arsitektur framework dan runtime, bukan dari implementasi kode aplikasi.

2.3 Alat dan Parameter Benchmarking

Proses *load testing* dilakukan menggunakan **ApacheBench (ab) versi 2.3**, sebuah alat benchmark standar yang banyak digunakan pada penelitian performa server web. ApacheBench dipilih karena kestabilannya dalam menghasilkan beban simultan dan kemampuannya menyediakan metrik performa terperinci secara konsisten.
Konfigurasi benchmarking ditetapkan sebagai berikut:

a. **READ**:
   1. Jumlah permintaan (total requests): 1000
   2. Tingkat konkurensi (concurrency): 20
   3. Metode HTTP: GET

b. **CREATE**:
   1. Jumlah permintaan: 100
   2. Tingkat konkurensi: 10
   3. Metode HTTP: POST

Perbedaan jumlah request pada masing-masing skenario disesuaikan dengan tingkat kompleksitas operasi sehingga mencegah trivialisasi beban WRITE dan potensi bottleneck database yang dapat mendistorsi keseluruhan pengukuran.
Parameter performa yang direkam dalam penelitian ini mencakup:

a. **Requests per Second (RPS)** – menunjukkan throughput server.
b. **Mean Latency** – waktu tanggap rata-rata setiap permintaan.
c. **Median Latency (P50)** – nilai tengah latensi respons.
d. **P95 Latency** – latensi pada persentil ke-95 yang merepresentasikan kondisi *near worst-case*.
e. **P99 Latency** – latensi ekstrem yang sangat mendekati batas terburuk.
f. **Maximum Latency** – waktu respons terpanjang yang tercatat.
g. **Failed Requests** – jumlah kegagalan request selama pengujian.
h. **Transfer Rate** – kecepatan pengiriman data dari server ke klien.

2.4 Prosedur Eksperimen

Setiap skenario pengujian dilakukan melalui tahapan berikut:

a. Inisialisasi server pada framework yang diuji dengan konfigurasi default production mode.
b. Verifikasi endpoint API untuk memastikan respons valid dan uniform.
c. Menjalankan ApacheBench sesuai parameter skenario pengujian.
d. Mencatat seluruh metrik keluaran benchmark secara otomatis.
e. Mengulang skenario pengujian sebanyak minimal tiga kali untuk setiap framework guna mengurangi fluktuasi hasil dan mengambil nilai rata-rata sebagai hasil akhir pengukuran.

JURNAL KOMPUTER TEKNOLOGI INFORMASI SISTEM KOMPUTER
p-ISSN: 2962-3022 | e-ISSN: 2963-7104
Volume 4 No 3 Februari 2022
Hal: 1583-1590
http://ejurnal.lkpkaryaprima.id/index.php/juktisi
DOI: https://doi.org/10.62712/juktisi.v4i3.717

Antara setiap sesi pengujian dilakukan *restart service worker* untuk menghindari efek akumulasi memori atau cache internal yang berpotensi memberikan bias terhadap hasil benchmark.

2.5 Teknik Analisis Data

Data mentah hasil benchmarking dianalisis secara deskriptif kuantitatif. Nilai rata-rata setiap metrik dibandingkan antar-framework guna mengidentifikasi perbedaan performa throughput dan latensi. Rasio peningkatan performa dihitung dengan membandingkan nilai RPS dan latency median antara Webman dan Laravel Octane. Selain evaluasi numerik, penafsiran dilakukan dengan mengaitkan hasil performa terhadap karakteristik arsitektur internal masing-masing framework, mencakup mekanisme event-loop, sistem middleware, konteks dependency container, serta pola eksekusi *worker persistence*.

Analisis berfokus pada dua aspek utama, yaitu:

a. **Efisiensi throughput**, yang diwakili oleh nilai RPS dan Transfer Rate.
b. **Stabilitas latensi**, yang diwakili oleh distribusi respon P50, P95, dan P99.

Hasil analisis kemudian dikaitkan dengan literatur terdahulu untuk menilai konsistensi temuan penelitian terhadap teori arsitektur worker model dan desain framework event-driven.

2.6 Validitas dan Reproduksibilitas

Untuk menjaga validitas internal penelitian, seluruh pengujian dilakukan dalam satu mesin uji dengan konfigurasi konsisten serta endpoint bersifat identik secara fungsional. Penggunaan alat benchmarking standar dan parameter terbuka memungkinkan eksperimen ini direplikasi oleh peneliti lain pada lingkungan serupa. Keterbatasan penelitian meliputi penggunaan localhost sebagai media pengujian tanpa faktor jaringan eksternal, sehingga hasil performa merepresentasikan performa server murni tanpa latensi jaringan. Namun demikian, hal tersebut justru mendukung validitas pengukuran internal dan memungkinkan fokus pada evaluasi kemampuan komputasi framework secara objektif.

3. HASIL DAN PEMBAHASAN

Bab ini menyajikan hasil pengujian performa Laravel Octane (FrankenPHP) dan Webman (Workerman) pada dua skenario utama, yaitu operasi **READ (pengambilan 1000 baris data)** dan **CREATE (penyimpanan 1000 baris data)** melalui endpoint API berbasis worker model PHP. Parameter evaluasi meliputi throughput (*Requests per Second*), latensi respon (median dan P95), serta stabilitas permintaan.

3.1 Hasil Pengujian Operasi READ (1000 Row)

Skenario READ merepresentasikan kondisi *data retrieval intensive* yang umum pada API modern, seperti dashboard analitik, penarikan laporan, atau pemanggilan data masif oleh klien. Uji beban dilakukan dengan total 1000 permintaan pada tingkat konkurensi 20 menggunakan metode HTTP GET.

Tabel 1. Hasil Benchmark Operasi READ

Framework	Requests per Second (RPS)	Median Latency (P50)	P95 Latency
Laravel Octane	137 RPS	118 ms	344 ms
Webman	265 RPS	41 ms	221 ms

Berdasarkan Tabel 3.1, Webman menunjukkan kinerja throughput hampir **dua kali lipat** dibanding Laravel Octane. Dengan capaian 265 RPS, Webman unggul sekitar **1,93 kali** dibandingkan Laravel Octane yang menghasilkan 137 RPS. Dari sisi latensi, Webman juga memperlihatkan performa signifikan dengan median latency **41 ms**, jauh lebih rendah dibanding Laravel Octane yang mencapai **118 ms**. Pada metrik P95, Webman mencatat **221 ms** sementara Laravel Octane mencapai **344 ms**, mengindikasikan bahwa respons Webman lebih konsisten dan memiliki *tail latency* yang lebih rendah.Perbedaan ini menunjukkan bahwa Webman lebih efisien menangani beban baca masif dengan concurrency tinggi, yang sangat dipengaruhi oleh implementasi *event-loop non-blocking I/O* pada Workerman. Sementara itu, Laravel Octane masih membawa overhead internal berupa middleware stack, dependency injection container, serta overhead ORM laravel yang meskipun persisten di worker, tetap lebih kompleks dibanding arsitektur minimalis Webman.

3.2 Hasil Pengujian Operasi CREATE (1000 Row)

Skenario CREATE merepresentasikan beban transaksi tulis (*write-intensive workload*) seperti input data pelanggan, pencatatan transaksi pembayaran, atau sinkronisasi data layanan eksternal. Uji beban dilakukan dengan 100 permintaan POST berkonkurensi 10, di mana setiap permintaan melakukan insert batch terhadap 1000 baris data.

JURNAL KOMPUTER TEKNOLOGI INFORMASI SISTEM KOMPUTER
p-ISSN: 2962-3022 | e-ISSN: 2963-7104
Volume 4 No 3 Februari 2022
Hal: 1583-1590
http://ejurnal.lkpkaryaprima.id/index.php/juktisi
DOI: https://doi.org/10.62712/juktisi.v4i3.717

Tabel 2. Hasil Benchmark Operasi CREATE

Framework	Requests per Second (RPS)	Median Latency (P50)	P95 Latency
Laravel Octane	39.49 RPS	216 ms	340 ms
Webman	69.66 RPS	107 ms	295 ms

Hasil pada Tabel 3.2 kembali memperlihatkan keunggulan Webman. Framework ini mencatat throughput sebesar 69,66 RPS, sekitar 1,77 kali lebih tinggi dibanding Laravel Octane yang mencatat 39,49 RPS. Dari segi median latency, Webman hampir 50% lebih cepat dengan waktu respons 107 ms, sementara Laravel Octane memerlukan 216 ms. Pada metrik P95, selisih masih terlihat jelas meskipun tidak sedrastis pada median, yakni 295 ms pada Webman dibanding 340 ms pada Laravel Octane.

Performa CREATE secara umum lebih rendah dibanding READ pada kedua framework karena beban operasi tulis di database melibatkan mekanisme sinkronisasi disk, locking, dan validasi data. Namun demikian, Webman tetap mempertahankan keunggulan throughput dan latensi karena eksekusi event-driven memungkinkan pemanfaatan thread serta resource proses worker secara lebih efisien pada fase menunggu I/O database (I/O wait).

3.2 Analisis Perbandingan Throughput

Tabel 3. Perbandingan Rasio Throughput

Operasi	RPS Laravel Octane	RPS Webman	Rasio Keunggulan Webman
READ	137	265	1,93 ×
CREATE	39,49	69,66	1,77 ×

Tabel 3.3 menunjukkan bahwa Webman secara konsisten unggul pada kedua tipe beban pengujian. Keunggulan throughput tertinggi terjadi pada operasi READ, yang mengindikasikan efektivitas arsitektur non-blocking dalam melayani permintaan paralel dengan overhead minimum. Pada operasi CREATE, meskipun bottleneck utama berasal dari sisi database, Webman tetap memperlihatkan keunggulan signifikan, menandakan bahwa framework ini mengelola *task scheduling* dan *connection handling* secara lebih optimal.

Sebaliknya, Laravel Octane menampilkan kestabilan kinerja yang baik, tetapi kompleksitas ekosistem Laravel yang terdiri atas middleware, event dispatcher, ORM abstractions, dan dependency container menyebabkan tambahan biaya pemrosesan pada setiap request.

3.3 Analisis Latensi dan Stabilitas Respons

Tabel 4. Ringkasan Median Latency

Operasi	Laravel Octane (ms)	Webman (ms)	Penurunan Latensi
READ	118	41	65% lebih cepat
CREATE	216	107	50% lebih cepat

Tabel 3.4 menunjukkan bahwa Webman memiliki tingkat latensi median yang secara konsisten lebih rendah pada semua skenario pengujian. Penurunan latensi sebesar **65% pada operasi READ** dan **50% pada CREATE** mencerminkan dampak langsung dari desain event-loop Workerman yang meminimalkan overhead context switching serta mengoptimalkan pengelolaan input/output.

Laravel Octane tetap menunjukkan stabilitas tinggi dan tidak mengalami lonjakan latensi abnormal yang dapat mengindikasi kegagalan sistem. Namun, nilai P95 yang relatif lebih tinggi pada kedua operasi menunjukkan adanya latency tail yang dipengaruhi overhead internal framework saat beban meningkat, terutama akibat pemanggilan middleware dan pemrosesan dependency injection container.

3.5 Analisis Arsitektur dan Implikasi Praktis

Keunggulan utama Webman dapat ditelusuri pada arsitektur *event-loop C-level* dari Workerman yang mampu mengelola ribuan koneksi simultan dengan overhead minimal. Framework Webman hanya menambahkan lapisan ringan di atas Workerman, menjadikan eksekusi logika aplikasi lebih dekat dengan proses server asli.

Sebaliknya, Laravel Octane menghadirkan pendekatan hybrid: worker persistence dikombinasikan dengan ekosistem Laravel yang kompleks. Walaupun pendekatan ini sukses menurunkan overhead bootstrap PHP-FPM secara signifikan, tumpukan middleware, abstraction layer, dan layanan tambahan Laravel tetap menghasilkan latency tambahan dibandingkan Webman.

JURNAL KOMPUTER TEKNOLOGI INFORMASI SISTEM KOMPUTER
p-ISSN: 2962-3022 | e-ISSN: 2963-7104
Volume 4 No 3 Februari 2022
Hal: 1583-1590
http://ejurnal.lkpkaryaprima.id/index.php/juktisi
DOI: https://doi.org/10.62712/juktisi.v4i3.717

Dalam konteks implementasi praktis, hasil penelitian ini mengindikasikan bahwa:

a. **Webman sangat ideal untuk sistem API skala besar** yang membutuhkan throughput tinggi dan latensi rendah, seperti microservices, aggregator data, dan sistem real-time.
b. **Laravel Octane lebih sesuai untuk aplikasi kompleks** dengan kebutuhan domain logic yang rumit, integrasi third-party luas, dan keunggulan developer productivity, meskipun performanya berada di bawah framework ringan.

3.6 Pembahasan Umum

Hasil penelitian ini mengonfirmasi teori bahwa arsitektur event-driven non-blocking memberikan keuntungan performa signifikan dalam konteks API heavy-load. Webman memanfaatkan pendekatan ini secara optimal melalui Workerman, sedangkan Laravel Octane menerapkan model worker persistence namun tetap membawa beban abstraction framework besar. Dengan demikian, dapat disimpulkan bahwa pilihan framework backend PHP tidak hanya ditentukan oleh kemudahan pengembangan, tetapi juga harus mempertimbangkan tuntutan performa aktual di lingkungan produksi. Trade-off antara kecepatan eksekusi dan kelengkapan ekosistem menjadi faktor utama penentu pemilihan solusi backend modern.

4. KESIMPULAN

Penelitian ini membandingkan kinerja Laravel Octane dengan FrankenPHP dan Webman dengan Workerman dalam skenario operasi READ dan CREATE API berbasis PHP worker model. Hasil pengujian menunjukkan bahwa Webman secara konsisten memiliki keunggulan signifikan dalam throughput dan latensi pada kedua operasi. Pada pengujian READ data masif, Webman mencatat peningkatan throughput hingga hampir dua kali lipat dibanding Laravel Octane, disertai penurunan median latensi lebih dari 60%. Pola serupa juga terlihat pada skenario CREATE, di mana Webman mempertahankan keunggulan sekitar 1,7 kali lebih cepat dengan waktu respons median sekitar 50% lebih rendah. Temuan ini mengindikasikan bahwa arsitektur event-driven non-blocking pada Workerman memiliki efektivitas tinggi dalam memaksimalkan pemanfaatan sumber daya server, terutama untuk beban API yang bersifat paralel dan intensif basis data. Sebaliknya, Laravel Octane menunjukkan stabilitas operasional yang baik serta konsistensi performa meskipun masih dibatasi oleh kompleksitas internal ekosistem Laravel seperti middleware, dependency injection container, dan lapisan ORM. Implikasi hasil penelitian ini menekankan bahwa pemilihan framework PHP tidak semata ditentukan oleh kemudahan pengembangan, tetapi harus mempertimbangkan kebutuhan performa aktual aplikasi. Webman lebih direkomendasikan untuk sistem API bertrafik tinggi dan layanan mikro yang membutuhkan respons sangat cepat, sementara Laravel Octane tetap relevan untuk aplikasi berskala kompleks yang memprioritaskan produktivitas pengembang dan fitur ekosistem. Kontribusi utama studi ini terletak pada penyediaan bukti empiris komparatif yang memperkaya literatur terkait evaluasi performa framework PHP berbasis worker model. Penelitian lanjutan disarankan untuk mencakup pengujian operasi UPDATE dan DELETE, simulasi beban yang lebih besar, serta validasi performa pada lingkungan server produksi berbasis Linux.

REFERENCES

[1] S. Suherman, H. Hermansyah, and J. Syahpita, “Sistem Alarm Deteksi Gerak Berbasis IoT Menggunakan Sensor PIR dan ESP32 dengan Notifikasi Telegram Real-Time,” *J. Komput. Teknol. Inf. Sist. Inf.*, vol. 4, no. 2, pp. 1469–1476, 2025.

[2] Evi Triana, Ade Zulkarnain Hasibuan, Arnes Sembiring, and Yessi Fitri Annisah Lubis, “Prototipe Alat Pakan Ternak Ayam Otomatis Dua Sisi Berbasis Mikrokontroler,” *J. Komput. Teknol. Inf. dan Sist. Inf.*, vol. 1, no. 2, pp. 130–137, 2022, doi: 10.62712/juktisi.v1i2.46.

[3] C. INFORMATIKA, Yusuf Muharam, and Taufik Hidayat, “Pengembangan Aplikasi Back-End E-Commerce Menggunakan Rest Api Golang Untuk Optimalisasi Kinerja Server,” *Comput. | J. Inform.*, vol. 11, no. 01, pp. 7–13, 2024, doi: 10.55222/computing.v11i01.1479.

[4] R. Nurcahyo, S. Bachri, and G. N. Ridwan, “nalisa Perbandingan Kinerja Protokol HTTP Dan Websocket Dalam Pengelolaan Pemesanan Dan Validasi Tiket Digital Bioskop Secara Realtime.,” *J. Inf. Technol. Innov.*, vol. 1, no. 2, 2023.

[5] M. A. Yaqin, “Pengembangan Aplikasi Marketplace Ikan Di Kabupaten Probolinggo Berbasis Frontend Backend Menggunakan React Js,” *NJCA (Nusantara J. Comput. Its Appl.*, vol. 8, no. 2, p. 63, 2023, doi: 10.36564/njca.v8i2.342.

[6] A. Nur *et al.*, “Landasan Standar Akuntansi Mengenai Perbandingan Prinsip KerangkaKonseptual Global Dan Realitas Indonesia,” *J. A.N.C Account. Tax Audit Bus. Inf. Syst. Informatics Technol.*, vol. 1, no. 3, pp. 200–215, 2025, [Online]. Available: https://journal.anc-aryantonurconsulting.com/tp

[7] E. Sutrisno, L. Safitri, L. P. Lestari, R. M. Akbar, and Y. N. Sukmaningtyas, “Pemanfaatan Teknologi Digital untuk Optimalisasi Pelayanan dan Dokumentasi Administratif melalui Web Desa Kentong, Lamongan,” *J. Ragam Pengabdi.*, vol. 2, no. 2, pp. 158–166, 2025.

[8] A. P. Kejora and W. N. Purwanti, “Efektivitas Penggunaan Node Js Dalam Pembuatan Rest Api Untuk Aplikasi

JURNAL KOMPUTER TEKNOLOGI INFORMASI SISTEM KOMPUTER
p-ISSN: 2962-3022 | e-ISSN: 2963-7104
Volume 4 No 3 Februari 2022
Hal: 1583-1590
http://ejurnal.lkpkaryaprima.id/index.php/juktisi
DOI: https://doi.org/10.62712/juktisi.v4i3.717

Katastrofa," in *Prosiding Seminar Nasional Sains dan Teknologi "SainTek,"* 2025, pp. 995–1008. [Online]. Available: https://conference.ut.ac.id/index.php/saintek/article/view/5135

[9] S. S. Raweyai and I. R. Widiasari, "Performance Testing of Academic Website Using Load Testing Method Supported By Apache Jmetertm At Xyz University," *J. Tek. Inform.*, vol. 5, no. 3, pp. 721–730, 2024, doi: 10.52436/1.jutif.2024.5.3.1796.

[10] A. Hadiewijaya and B. Wasito, "Perbandingan Efisiensi Waktu dan Memori Pada C# dan Java dengan Metode Benchmarking," *bit-Tech*, vol. 7, no. 2, pp. 554–560, 2024, doi: 10.32877/bt.v7i2.1906.

[11] R. Anggarah *et al.*, "Studi Perbandingan Penerapan Pola Model-View-Controller (MVC) dalam Lima Framework Web Populer: Laravel, Django, Ruby on Rails, ASP.NET MVC, Spring MVC," *Indones. J. Comput. Sci. Eng.*, vol. 2, no. 1, pp. 16–22, 2025.

[12] Z. M. Anggraini, "Pengujian CRUD Link Pada Aplikasi LinkStack Berbasis Web Menggunakan Black Box Testing," *J. Inf. Syst. Bus. Technol.*, vol. 1, no. 1, pp. 83–90, 2025.

[13] A. S. Azzahidi, B. Wijayanto, and A. Darmawan, "Performance Evaluation of Backend Frameworks for REST API: A Comparative Study of Spring Boot, Flask, Express.js, Laravel FrankenPHP, and Gin," *J. Tek. Inform.*, vol. 6, no. 4, pp. 2405–2419, 2025, doi: 10.52436/1.jutif.2025.6.4.4811.

[14] D. J. Riyanto, P. Pizaini, N. S. H., and M. Affandes, "Implementasi Service Choreography Pattern Arsitektur Microservice Classroom Akademik Menggunakan Docker," *JIPI (Jurnal Ilm. Penelit. dan Pembelajaran Inform.*, vol. 7, no. 3, pp. 768–779, 2022, doi: 10.29100/jipi.v7i3.3126.

[15] A. Valerian, D. Bernady, F. Wahyudi, J. Pratama Dinatha, and O. Dhylan Barletyano, "Implementasi Unit Testing, Integration Testing, System Testing, Dan Validation Testing Pada Aplikasi Berbasis Website," *JATI (Jurnal Mhs. Tek. Inform.*, vol. 9, no. 4, pp. 6354–6362, 2025, doi: 10.36040/jati.v9i4.14067.