**Jurnal Kreativitas**
**Teknologi dan Komputer** Vol. 17 No.1, Januari 2026

# ANALISA PERBANDINGAN PERFORMA PHP PADA WEB SERVICE: SWOOLE VS APACHE VS NGINX VS PHP-FPM

**Sabariman[1], Cindy[2]**
Universitas Internasional Batam
E-mail: sabariman@uib.ac.id[1], 2232055.cindy@uib.edu[2]

**ABSTRAK**

Penelitian ini bertujuan untuk menganalisis dan membandingkan performa web service PHP pada empat konfigurasi server, yaitu Apache, Nginx, PHP-FPM, dan Swoole, dalam lingkungan cloud computing. Pengujian dilakukan menggunakan pendekatan eksperimental pada platform Amazon Web Services (AWS) dengan spesifikasi server yang seragam. Simulasi beban dilakukan menggunakan k6 dengan variasi jumlah Virtual Users hingga 5000 VUs untuk merepresentasikan skenario akses pengguna secara simultan. Parameter performa yang dianalisis meliputi response time, latency, throughput, error rate, serta penggunaan sumber daya sistem berupa CPU dan memori. Hasil penelitian menunjukkan bahwa Swoole memberikan performa paling unggul dan stabil dibandingkan konfigurasi web service PHP lainnya, khususnya pada skenario concurrency tinggi. Arsitektur asynchronous dan penggunaan coroutine memungkinkan Swoole mempertahankan response time dan latency yang rendah, throughput yang tinggi, serta error rate yang relatif stabil meskipun jumlah Virtual Users meningkat hingga 5000 VUs. Selain itu, Swoole juga menunjukkan efisiensi penggunaan sumber daya CPU dan memori yang lebih baik, sehingga mampu menangani lonjakan beban secara optimal tanpa degradasi kinerja yang signifikan. Temuan ini menegaskan bahwa Swoole merupakan solusi web service PHP yang paling optimal untuk lingkungan cloud dengan kebutuhan skalabilitas dan performa tinggi.

**Kata Kunci:** Web Service PHP, Cloud Computing, Swoole, Apache, Nginx.

***ABSTRACT***

*This study aims to analyze and compare the performance of PHP web services on four server configurations, namely Apache, Nginx, PHP-FPM, and Swoole, in a cloud computing environment. Testing was conducted using an experimental approach on the Amazon Web Services (AWS) platform with uniform server specifications. Load simulations were conducted using k6 with varying numbers of Virtual Users up to 5000 VUs to represent simultaneous user access scenarios. Performance parameters analyzed included response time, latency, throughput, error rate, and system resource usage in the form of CPU and memory. The results of the study show that Swoole provides the most superior and stable performance compared to other PHP web service configurations, especially in high concurrency scenarios. The asynchronous architecture and the use of coroutines allow Swoole to maintain low response time and latency, high throughput, and a relatively stable error rate even when the number of Virtual Users increases to 5000 VUs. In addition, Swoole also shows better efficiency in the use of CPU and memory resources, so it is able to handle load spikes optimally without significant performance degradation. These findings confirm that Swoole is the most optimal PHP web service solution for cloud environments with high scalability and performance requirements.*

***Keywords:*** *Web Service PHP, Cloud Computing, Swoole, Apache, Nginx.*

# 1. PENDAHULUAN

Perkembangan teknologi cloud computing telah mendorong transformasi besar dalam penyediaan layanan digital, khususnya pada pengelolaan website yang kini semakin mengandalkan infrastruktur berbasis cloud untuk memenuhi kebutuhan skalabilitas, ketersediaan layanan, dan performa tinggi (Saleh, 2025). Lonjakan jumlah pengguna internet serta meningkatnya kompleksitas aplikasi web modern membuat tuntutan terhadap kecepatan respons dan stabilitas sistem menjadi semakin tinggi (Amrullah et al., 2023). Dalam konteks tersebut, performa web server memegang peranan sentral karena berfungsi sebagai komponen yang memproses setiap permintaan pengguna (Amin, 2025). Website yang tidak mampu merespons secara cepat cenderung menurunkan kualitas pengalaman pengguna dan berdampak langsung pada kepercayaan serta keberlanjutan layanan digital. Oleh sebab itu, analisis performa web server berbasis cloud menjadi urgensi yang semakin relevan untuk dikaji melalui pendekatan ilmiah.

Dalam arsitektur cloud modern, web service bertindak sebagai penghubung utama antara pengguna dengan aplikasi melalui protokol standar. Seiring berkembangnya teknologi, web service tidak hanya memuat konten statis tetapi juga menjalankan transaksi real-time, pemrosesan data dinamis, integrasi API lintas platform, serta mendukung arsitektur microservices (Harris et al., 2020). Karakteristik ini menuntut web server untuk memiliki kemampuan penanganan permintaan secara konkuren, efisien, dan stabil, terutama dalam lingkungan cloud yang skalanya dapat bertambah secara cepat (Amin, 2025). Performa sistem web sangat dipengaruhi oleh arsitektur web server yang digunakan, mulai dari model process-based pada Apache, event-driven pada Nginx, manajemen proses terpisah melalui PHP-FPM, hingga arsitektur asynchronous I/O seperti Swoole yang menawarkan efisiensi lebih tinggi dengan dukungan coroutine (Mufid et al., 2024) (Herman et al., 2025).


Gambar 1. Arsitektur Web Server

Gambar 1. Arsitektur Web Server menggambarkan arsitektur pengujian performa web service PHP yang dilakukan dalam lingkungan cloud terkontrol. Alur dimulai dari website request yang merepresentasikan permintaan pengguna ke aplikasi web, kemudian diteruskan ke lapisan web service sebagai pengelola utama permintaan. Untuk mensimulasikan kondisi penggunaan nyata dengan tingkat concurrency tinggi, permintaan dikirim secara paralel menggunakan k6 sebagai alat load testing. Seluruh proses pengujian dijalankan dalam cloud environment berbasis AWS (Amazon Web Service), sehingga setiap konfigurasi server memiliki spesifikasi dan kondisi infrastruktur yang sama. Pada lapisan inti, request diproses oleh empat konfigurasi web server yang berbeda, yaitu Apache dengan arsitektur process-based, Nginx dengan arsitektur event-driven, PHP-FPM sebagai manajemen proses PHP terpisah, serta Swoole yang menggunakan pendekatan asynchronous dan coroutine. Perbedaan arsitektur ini menjadi fokus utama penelitian karena memengaruhi cara server menangani permintaan secara simultan. Hasil pemrosesan kemudian dianalisis berdasarkan metrik performa utama, yaitu response time, throughput, serta penggunaan CPU dan

memori, yang digunakan sebagai dasar perbandingan objektif untuk menilai efektivitas masing-masing web server dalam menangani beban kerja pada lingkungan cloud.

Berbagai penelitian terdahulu menunjukkan bahwa web server berarsitektur event-driven dan asynchronous cenderung memberikan kinerja yang lebih baik pada beban tinggi dibanding pendekatan tradisional (Amin, 2025); (Harris et al., 2020). Selain itu, performa web server juga dipengaruhi oleh lingkungan komputasi seperti virtual machine, container, serta platform cloud yang digunakan (Amrullah et al., 2023). Perbedaan alat uji seperti Apache Benchmark, JMeter, dan k6 juga dapat menghasilkan variasi hasil performa, sehingga pengujian yang metodologis dan terstandarisasi menjadi sangat penting untuk menghasilkan kesimpulan yang valid (Amin, 2025) (Saleh, 2025) (Niarman, 2023). Hal ini menunjukkan adanya ruang penelitian yang signifikan untuk membandingkan performa web service PHP secara komprehensif pada berbagai web server modern dalam kondisi cloud yang terkontrol.

Penelitian mengenai performa web service telah menunjukkan bahwa arsitektur server merupakan faktor utama yang menentukan efisiensi pemrosesan permintaan, terutama pada aplikasi berbasis cloud yang menangani trafik tinggi. Studi yang dilakukan oleh Fitriana dan Seimahuira (2023) dan Amin (2025) membuktikan bahwa server dengan arsitektur event-driven seperti Nginx dan LiteSpeed menunjukkan performa lebih unggul dalam hal response time dan throughput dibandingkan Apache yang berbasis proses. Keunggulan arsitektur event-driven disebabkan oleh kemampuan menangani ribuan koneksi secara simultan tanpa harus membuat thread atau proses baru untuk setiap permintaan, sehingga overhead server dapat ditekan secara signifikan. Sementara itu, server berbasis proses seperti Apache tetap memiliki kelebihan pada stabilitas modul dan fleksibilitas konfigurasi, tetapi kecenderungannya mengalami bottleneck pada beban yang sangat tinggi. Temuan tersebut memberikan indikasi awal bahwa perbedaan arsitektur menjadi aspek fundamental yang perlu dianalisis lebih mendalam ketika mengevaluasi performa web service berbasis PHP.

Selain arsitektur server, beberapa penelitian menekankan pentingnya platform dan lingkungan eksekusi dalam menentukan performa aplikasi web. Amrullah et al. (2023) menunjukkan bahwa performa web server dapat berbeda signifikan antara platform cloud yang berbeda, seperti Azure dan AWS, meskipun menggunakan konfigurasi server yang serupa. Penelitian lain oleh Ramadhan dan Solehudin (2024) dan Zuril (2024) menegaskan bahwa penggunaan container seperti Docker dapat meningkatkan fleksibilitas deployment, namun sering kali menambah overhead yang berpotensi menurunkan performa server. Hal ini menunjukkan bahwa pemilihan platform cloud, konfigurasi virtual machine, serta penggunaan kontainerisasi dapat memengaruhi hasil pengujian performa secara signifikan. Oleh karena itu, penelitian yang dilakukan dalam lingkungan cloud terkontrol seperti AWS menjadi penting untuk memastikan hasil yang lebih akurat dan representatif.

Studi terkait eksekusi PHP juga memberikan kontribusi penting dalam memahami performa web service. Beberapa penelitian sebelumnya menunjukkan bahwa model synchronous tradisional pada PHP-FPM memiliki keterbatasan dalam menangani beban concurrency tinggi. Sebaliknya, penelitian yang dilakukan oleh Haryani et al. (2024) membuktikan bahwa Swoole sebagai extension PHP dengan mekanisme asynchronous dan coroutine mampu memberikan peningkatan performa signifikan pada aplikasi real-time. PHP dengan Swoole terbukti memiliki throughput lebih tinggi dan latensi lebih rendah dibandingkan model eksekusi PHP konvensional, bahkan dalam skenario ribuan koneksi bersamaan. Hal ini memperlihatkan bahwa teknologi asynchronous pada PHP menjadi faktor potensial untuk meningkatkan performa aplikasi berbasis cloud.

Di samping itu, penelitian lain seperti Perdana dan Prihanto (2025), Prasetyo et al. (2025), Jiwandono (2021), Aziz (2025), Endra et al. (2022), Firmansyah dan Sudarmilah (2024) menunjukkan bahwa setiap web server memiliki karakteristik performa yang berbeda

pada tingkat beban tertentu. Lighttpd misalnya, unggul dalam konsumsi resource namun tidak optimal pada beban berat, sementara Apache cenderung stabil pada beban ringan tetapi mengalami penurunan kinerja saat concurrency meningkat. Temuan-temuan tersebut menegaskan bahwa tidak ada satu server yang dominan di semua aspek, sehingga perbandingan performa harus dilakukan secara komprehensif dengan memperhatikan berbagai parameter seperti latency, throughput, error rate, dan penggunaan CPU/RAM.

Penelitian ini dilaksanakan untuk menjawab keterbatasan yang masih ditemukan dalam studi-studi sebelumnya terkait evaluasi performa web server berbasis PHP. Sejumlah penelitian terdahulu umumnya hanya membandingkan dua atau tiga jenis web server, menggunakan lingkungan lokal atau on-premise, serta berfokus pada satu indikator kinerja tertentu seperti response time atau throughput saja. Selain itu, penggunaan alat uji beban modern yang mampu merepresentasikan skenario akses pengguna secara realistis pada lingkungan cloud masih relatif terbatas.

Atas dasar itu, maka dalam studi ini akan dilakukan analisis komprehensif terhadap performa PHP pada empat konfigurasi web server, yaitu Apache, Nginx, PHP-FPM, dan Swoole, dengan memanfaatkan AWS sebagai platform cloud computing. Pengujian dilakukan menggunakan k6 sebagai alat simulasi beban untuk merepresentasikan variasi tingkat permintaan pengguna secara terukur dan konsisten. Evaluasi kinerja tidak hanya difokuskan pada response time dan throughput, tetapi juga mencakup konsumsi sumber daya sistem, khususnya CPU dan memori, yang sering kali belum dianalisis secara terintegrasi dalam penelitian.

## 2. METODE PENELITIAN

Penelitian ini menggunakan pendekatan eksperimental (experimental research) dengan tujuan melakukan pengujian langsung terhadap performa empat web service berbasis PHP, yaitu Apache, Nginx, PHP-FPM, dan Swoole, pada lingkungan cloud AWS. Pendekatan eksperimen dipilih karena memungkinkan peneliti melakukan kontrol penuh terhadap variabel-variabel teknis seperti spesifikasi server, konfigurasi web service, traffic load, serta parameter pengujian sehingga hasil dapat dibandingkan secara objektif. Seluruh pengujian dilakukan dalam kondisi lingkungan yang terkontrol untuk memastikan bahwa perbedaan performa hanya dipengaruhi oleh karakteristik web service yang diuji.

Subjek penelitian berupa aplikasi PHP sederhana (benchmark script) yang memiliki fungsi komputasi dasar dan tidak melibatkan operasi eksternal seperti akses database. Penggunaan aplikasi sederhana bertujuan untuk mengisolasi performa asli web server sehingga tidak dipengaruhi faktor lain seperti framework atau I/O eksternal. Objek pengujian terdiri dari empat konfigurasi server: Apache + PHP-FPM, Nginx + PHP-FPM, Nginx standalone, dan Swoole asynchronous server.

Tabel 1. Instrumen Penelitian

| No | Instrumen Penelitian | Fungsi Utama | Keterangan |
|---|---|---|---|
| 1 | *AWS EC2* | Lingkungan *server* pengujian | Digunakan sebagai platform *cloud* untuk menjalankan seluruh *web service PHP*. Spesifikasi *instance* disamakan pada setiap skenario pengujian guna memastikan hasil pengukuran performa yang objektif dan terkontrol. |

| | | | |
|---|---|---|---|
| 2 | *k6* | *Performance testing tool* | Digunakan untuk melakukan *load test* dan *stress test*. *k6* mampu mensimulasikan beban tinggi hingga ribuan *Virtual Users (VU)* secara efisien dengan arsitektur *event-loop asynchronous*. |
| 3 | *Netdata* | Monitoring sistem | Digunakan untuk memantau dan mencatat metrik performa server, meliputi penggunaan *CPU*, memori, dan *I/O* selama proses pengujian berlangsung. |
| 4 | *Script* otomatisasi *deployment* | Standarisasi konfigurasi server | Digunakan untuk mengotomatisasi proses instalasi dan konfigurasi *web service* pada setiap server agar seluruh lingkungan pengujian memiliki konfigurasi yang identik. |

Tabel 1. Instrumen Penelitian menjelaskan instrumen penelitian yang digunakan dalam pengujian performa web service PHP beserta fungsi dan keterangannya. AWS EC2 digunakan sebagai lingkungan server pengujian yang berperan sebagai platform cloud untuk menjalankan seluruh web service PHP, dengan spesifikasi instance yang disamakan pada setiap skenario agar hasil pengukuran performa bersifat objektif dan terkontrol. Instrumen kedua adalah k6, yaitu performance testing tool yang digunakan untuk melakukan load test dan stress test, di mana k6 mampu mensimulasikan beban tinggi hingga ribuan Virtual Users (VU) secara efisien berkat arsitektur event-loop asynchronous. Selanjutnya, Netdata dimanfaatkan sebagai alat monitoring sistem untuk memantau dan mencatat metrik performa server, seperti penggunaan CPU, memori, dan I/O, selama proses pengujian berlangsung. Instrumen terakhir adalah script otomatisasi deployment yang berfungsi untuk melakukan standarisasi konfigurasi server, sehingga proses instalasi dan pengaturan web service pada setiap server dapat dilakukan secara otomatis dan identik, guna meminimalkan bias konfigurasi dalam hasil pengujian.

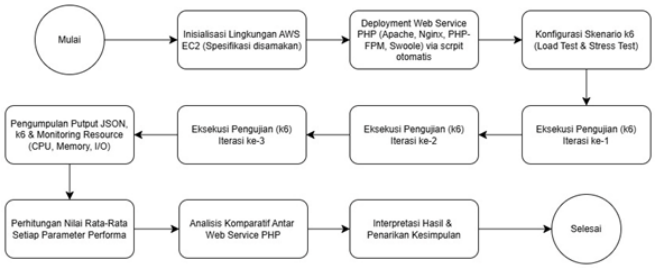
Tabel 2. Parameter Performa yang Diuji

| **No** | **Parameter Performa** | **Definisi** | **Satuan Ukur** | **Alat Ukur** |
|---|---|---|---|---|
| 1 | *Response Time* | Waktu total yang dibutuhkan *server* untuk merespons permintaan sejak *request* dikirim hingga *response* diterima sepenuhnya | Milidetik (ms) | *k6* |
| 2 | *Latency* | Waktu tunda awal *(delay)* antara pengiriman *request* dan penerimaan *byte* pertama dari server | Milidetik (ms) | *k6* |
| 3 | *Throughput* | Jumlah permintaan yang berhasil diproses server dalam satu detik | *Request per Second (RPS)* | *k6* |

| | | | | |
|---|---|---|---|---|
| 4 | *Error Rate* | Persentase permintaan yang gagal diproses oleh server selama pengujian | Persentase (%) | *k6* |
| 5 | *Resource Usage* | Tingkat penggunaan sumber daya server selama pengujian berlangsung, meliputi *CPU*, memori, dan *I/O* | Persentase (%) / MB | *Netdata* |

Tabel 2. Parameter Performa yang Diuji menjelaskan parameter performa yang digunakan dalam pengujian web service beserta definisi, satuan ukur, dan alat pengukurnya. Parameter pertama adalah response time, yaitu waktu total yang dibutuhkan server untuk merespons permintaan mulai dari request dikirim hingga response diterima secara lengkap oleh klien, yang diukur dalam milidetik menggunakan k6. Parameter kedua, latency, menggambarkan waktu tunda awal antara pengiriman request dan diterimanya byte pertama dari server, sehingga mencerminkan kecepatan awal respons sistem, juga diukur dalam milidetik dengan k6. Selanjutnya, throughput menunjukkan kemampuan server dalam memproses permintaan, yang diukur berdasarkan jumlah request yang berhasil ditangani per detik (RPS) menggunakan k6. Parameter keempat adalah error rate, yaitu persentase permintaan yang gagal diproses selama pengujian berlangsung, yang mencerminkan tingkat keandalan sistem dan diukur dalam persen dengan k6. Parameter terakhir adalah resource usage, yang menggambarkan tingkat pemanfaatan sumber daya server selama pengujian, mencakup penggunaan CPU, memori, dan I/O, yang diukur dalam satuan persentase atau megabyte menggunakan Netdata. Keseluruhan parameter ini digunakan untuk memberikan gambaran menyeluruh mengenai kinerja, efisiensi, dan stabilitas web service yang diuji.

Desain eksperimen dilakukan dengan menjalankan skenario load test dan stress test. Skenario load test digunakan untuk mengamati performa server pada beban bertahap menggunakan struktur stages pada k6, sedangkan stress test dilakukan untuk mengukur kemampuan maksimum server hingga mencapai titik kegagalan (failure point) pada 5000 VU sesuai metode pada landasan teori




Gambar 2. Alur Teknik Pengambilan Data

Gambar 2. Alur Teknik Pengambilan Data menunjukkan alur metodologi pengujian performa web service PHP yang dilakukan secara sistematis dari awal hingga akhir. Proses dimulai dengan tahap Mulai, dilanjutkan dengan inisialisasi lingkungan AWS EC2 di mana spesifikasi server disamakan untuk memastikan keadilan dan konsistensi pengujian. Selanjutnya dilakukan deployment web service PHP menggunakan berbagai teknologi server, seperti Apache, Nginx, PHP-FPM, dan Swoole, yang diotomatisasi melalui skrip agar konfigurasi berjalan seragam. Setelah sistem siap, peneliti melakukan konfigurasi skenario pengujian menggunakan k6 yang mencakup load test dan stress test. Pengujian kemudian dieksekusi sebanyak tiga iterasi untuk meningkatkan reliabilitas hasil. Pada setiap pengujian, data keluaran k6 dikumpulkan dalam format JSON disertai monitoring penggunaan sumber daya server, meliputi CPU, memori, dan I/O jaringan. Data yang diperoleh selanjutnya digunakan untuk menghitung nilai rata-rata setiap parameter performa, seperti response time, throughput, dan error rate. Tahap berikutnya adalah analisis komparatif antar web service

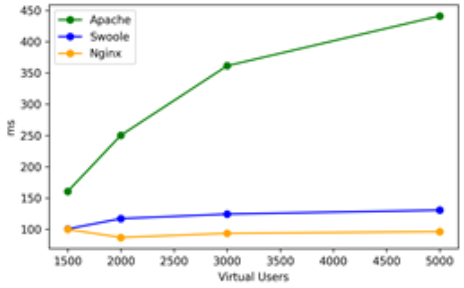
PHP untuk mengidentifikasi perbedaan kinerja masing-masing teknologi. Hasil analisis tersebut kemudian diinterpretasikan secara menyeluruh guna menarik kesimpulan mengenai performa terbaik dan karakteristik setiap web service, hingga akhirnya proses penelitian dinyatakan selesai.

Teknik pengambilan data dilakukan secara otomatis melalui script k6 yang mengirimkan permintaan HTTP secara paralel ke masing-masing web service. Data dikumpulkan mencakup waktu respons per request, total request sukses, request gagal, puncak latency, throughput, serta penggunaan sumber daya server. Semua data kemudian direkap dan dianalisis menggunakan metode perbandingan kuantitatif untuk menentukan keunggulan relatif setiap web server. Analisis dilakukan dengan menghitung selisih performa antar server serta grafik tren kenaikan beban terhadap kestabilan server. Hasil analisis digunakan untuk menjawab rumusan masalah pada penelitian, yaitu menentukan web service PHP yang paling optimal pada lingkungan cloud.

Metode penelitian ini secara keseluruhan dirancang agar dapat merepresentasikan perilaku web server PHP dalam kondisi nyata di lingkungan cloud, sehingga hasilnya dapat dijadikan rujukan bagi pengembang dan praktisi IT dalam memilih konfigurasi web service yang paling efisien. Pendekatan eksperimental ini juga memungkinkan adanya evaluasi objektif terhadap perbedaan arsitektur server seperti process-based (Apache), event-driven (Nginx), dan asynchronous coroutine (Swoole), yang merupakan fokus utama penelitian.

## 3. HASIL DAN PEMBAHASAN

Gambar 3. Perbandingan Average duration merepresentasikan efisiensi pemrosesan request secara keseluruhan, yang mencakup waktu antrean, eksekusi aplikasi, dan pengiriman respons. Nilai average duration yang rendah menunjukkan bahwa sistem mampu memproses permintaan secara cepat tanpa penumpukan antrean yang signifikan. Sebaliknya, peningkatan average duration mengindikasikan terjadinya ketidakseimbangan antara laju kedatangan request dan kapasitas pemrosesan server, yang umumnya disebabkan oleh meningkatnya queueing delay ketika thread, process, atau event loop sedang sibuk. Pada sistem dengan overhead eksekusi tinggi atau model pemrosesan bersifat blocking, kondisi ini dapat menyebabkan antrean request semakin panjang dan average duration meningkat tajam. Sebaliknya, kestabilan average duration mencerminkan kemampuan sistem dalam mengelola request secara efisien, yang didukung oleh mekanisme seperti non-blocking I/O, asynchronous processing, dan coroutine.




Gambar 3. Perbandingan Average Duration

Gambar 3. Perbandingan Average Duration menunjukkan perbedaan yang jelas antara Apache, Nginx, dan Swoole dalam menangani peningkatan beban Virtual Users (VUs). Pada beban awal 1500 VUs, Swoole mencatat average duration terendah di kisaran 100 ms, diikuti oleh Nginx sekitar 99 ms, sementara Apache menunjukkan nilai tertinggi sekitar 160 ms. Kondisi ini mengindikasikan bahwa sejak beban awal, Swoole telah mampu memproses request secara lebih efisien dibandingkan Nginx dan Apache.

Ketika beban meningkat menjadi 2000 VUs, grafik menunjukkan bahwa average duration Swoole naik secara moderat ke sekitar 117 ms, sementara Nginx berada sedikit

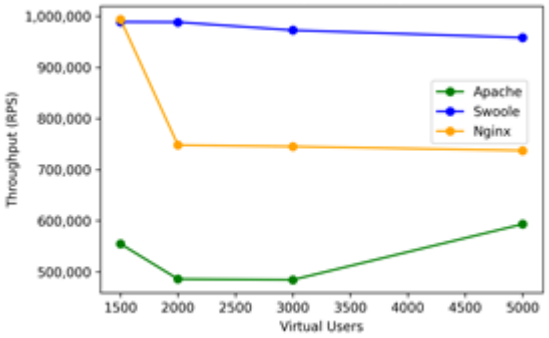
lebih rendah di kisaran 86 ms, dan Apache meningkat signifikan hingga sekitar 250 ms. Pada titik ini, Nginx masih mampu mempertahankan waktu pemrosesan yang sedikit lebih rendah dibandingkan Swoole, sedangkan Apache mulai mengalami peningkatan antrean request akibat keterbatasan arsitektur berbasis proses.

Pada beban 3000 VUs, average duration Swoole meningkat ke sekitar 124 ms dan terlihat lebih tinggi dibandingkan Nginx yang berada di kisaran 93 ms, sementara Apache melonjak tajam hingga sekitar 361 ms. Hal ini menunjukkan bahwa pada tingkat concurrency menengah, Nginx masih mampu menjaga efisiensi waktu pemrosesan sedikit lebih baik dibandingkan Swoole, meskipun perbedaannya relatif kecil. Namun demikian, jarak performa antara Swoole dan Apache semakin melebar, menandakan bahwa arsitektur asynchronous Swoole tetap jauh lebih efisien dibandingkan model eksekusi Apache.

Pada beban maksimum 5000 VUs, average duration Swoole berada di kisaran 130 ms, sedangkan Nginx sedikit lebih rendah di sekitar 96 ms, dan Apache mencapai nilai tertinggi sekitar 441 ms. Pada kondisi ini, baik Swoole maupun Nginx masih menunjukkan peningkatan yang relatif terkendali, sementara Apache mengalami degradasi performa yang sangat signifikan akibat penumpukan antrean request.

Secara keseluruhan, Gambar 3. Perbandingan Average Duration menunjukkan bahwa perbedaan pola kenaikan dan kestabilan waktu pemrosesan sangat dipengaruhi oleh arsitektur internal masing-masing web service. Apache mengalami peningkatan average duration yang paling tajam seiring bertambahnya jumlah Virtual Users, yang mengindikasikan terjadinya penumpukan antrean request akibat model eksekusi berbasis proses atau thread yang bersifat blocking dan memiliki overhead tinggi. Sebaliknya, Swoole menunjukkan kenaikan average duration yang lebih gradual dan terkendali, mencerminkan kemampuan arsitektur asynchronous dan coroutine dalam mengelola concurrency tinggi tanpa memperpanjang antrean secara signifikan. Nginx secara konsisten mencatat average duration terendah pada hampir seluruh tingkat beban, yang menunjukkan efisiensi event-driven dalam menangani request ringan; namun keunggulan ini perlu dipertimbangkan bersama metrik lain seperti error rate dan throughput. Dengan demikian, stabilitas average duration pada Swoole menegaskan bahwa mekanisme non-blocking dan proses long-running lebih efektif dalam menjaga efisiensi pemrosesan pada beban tinggi dibandingkan arsitektur tradisional berbasis proses seperti Apache.

Gambar 4. Perbandingan Throughput menggambarkan kapasitas aktual sistem dalam memproses permintaan per satuan waktu. Nilai throughput yang tinggi menunjukkan bahwa server mampu memanfaatkan sumber daya secara optimal untuk menghasilkan output maksimum. Peningkatan throughput biasanya terjadi ketika mekanisme concurrency berjalan efisien dan resource seperti CPU serta memory masih tersedia dalam jumlah cukup. Namun, throughput tidak selalu meningkat seiring bertambahnya permintaan. Ketika sistem mencapai saturation point, penambahan request tidak lagi meningkatkan output, melainkan justru menyebabkan stagnasi atau penurunan throughput. Kondisi ini terjadi akibat meningkatnya overhead internal seperti context switching, contention antar proses, atau locking pada resource bersama. Penurunan throughput juga sering berkorelasi dengan meningkatnya error rate. Ketika sebagian request gagal diproses akibat timeout atau resource exhaustion, jumlah request sukses otomatis menurun. Oleh karena itu, throughput merupakan indikator penting untuk mengidentifikasi batas kapasitas sistem dan efisiensi pemrosesan secara keseluruhan.




Gambar 4. Perbandingan Throughput

Gambar 4. Perbandingan Throughput awalnya menunjukkan bahwa Nginx secara menghasilkan throughput tertinggi dibandingkan Swoole dan Apache pada seluruh tingkat beban Virtual Users. Pada 1500 VUs, throughput Nginx sekitar 993.000 RPS, sedangkan Swoole berada di kisaran 988.000 RPS dan Apache hanya sekitar 554.000 RPS. Dengan demikian, throughput Nginx sekitar 0.5% lebih tinggi dibanding Swoole dan 79% lebih tinggi dibanding Apache, yang menandakan efisiensi pemrosesan request yang lebih baik pada beban awal.

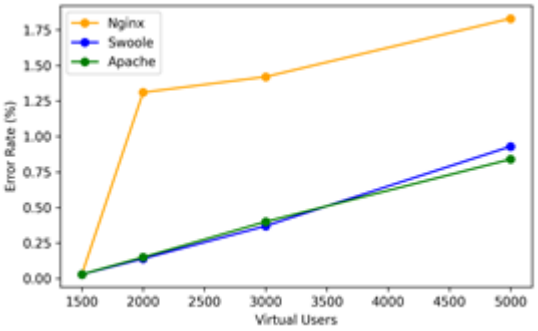
Ketika beban meningkat ke 2000–3000 VUs, throughput Swoole relatif stabil di kisaran 988.000–972.000 RPS, sementara Nginx mengalami penurunan signifikan ke sekitar 747.000 RPS dan Apache turun ke kisaran 485.000 RPS. Pada kondisi ini, throughput Swoole sekitar 31% lebih tinggi dibanding Nginx dan 102% lebih tinggi dibanding Apache, menunjukkan bahwa Swoole mampu mempertahankan kapasitas pemrosesan meskipun concurrency meningkat.

Pada beban maksimum 5000 VUs, throughput Swoole masih berada di sekitar 958.000 RPS, sedangkan Nginx berada di kisaran 737.000 RPS dan Apache sekitar 593.000 RPS. Artinya, throughput Swoole sekitar 30% lebih tinggi dibanding Nginx dan 61% lebih tinggi dibanding Apache.

Secara keseluruhan, perbedaan pola throughput pada masing-masing web service sangat dipengaruhi oleh arsitektur pemrosesan request yang digunakan. Nginx mampu menghasilkan throughput tertinggi pada beban awal karena arsitektur event-driven yang sangat efisien dalam menangani koneksi dan request ringan. Namun, ketika jumlah Virtual Users meningkat, throughput Nginx menurun secara signifikan akibat keterbatasan pada sisi backend, khususnya PHP-FPM, yang menyebabkan bottleneck meskipun lapisan web server masih mampu menangani koneksi dengan baik. Sebaliknya, Swoole menunjukkan throughput yang relatif stabil pada seluruh tingkat beban karena menggunakan model long-running process dan asynchronous I/O dengan dukungan coroutine, sehingga mampu memproses banyak request secara paralel tanpa overhead pembuatan proses atau thread baru. Apache, di sisi lain, mengalami throughput yang paling rendah dan fluktuatif karena model eksekusi berbasis proses atau thread yang memiliki overhead tinggi dan kurang efisien dalam menangani concurrency besar. Dengan demikian, kestabilan throughput Swoole pada beban menengah hingga tinggi menegaskan keunggulan arsitektur asynchronous dalam menjaga kapasitas pemrosesan, sementara penurunan throughput pada Nginx dan Apache mencerminkan keterbatasan arsitektural masing-masing dalam menghadapi lonjakan concurrency.

Gambar 5. Perbandingan Error rate menunjukkan tingkat reliabilitas sistem dalam mempertahankan layanan. Nilai error rate yang rendah menandakan bahwa sistem masih memiliki ruang kapasitas dan mampu menangani lonjakan permintaan tanpa kehilangan stabilitas. Sebaliknya, peningkatan error rate menjadi indikator awal bahwa sistem mulai mengalami tekanan serius. Error rate meningkat ketika server tidak mampu menyediakan resource yang dibutuhkan oleh request, seperti koneksi socket, worker, memory buffer, atau

waktu eksekusi CPU. Kondisi ini menyebabkan request mengalami timeout, connection reset, atau kegagalan pemrosesan. Pada beberapa sistem, error rate meningkat sebagai bentuk mekanisme proteksi, di mana server secara sengaja menolak request baru untuk menjaga kestabilan internal. Menariknya, peningkatan error rate tidak selalu diikuti oleh peningkatan latency. Dalam beberapa kasus, sistem justru mempertahankan latency rendah dengan cara menolak request lebih awal. Hal ini mencerminkan adanya trade-off antara kecepatan respons dan tingkat keberhasilan request, yang harus dipertimbangkan dalam evaluasi performa sistem.




Gambar 5. Perbandingan Error Rate

Gambar 5. Perbandingan Error Rate memperlihatkan perbedaan kemampuan masing-masing web service PHP dalam menjaga keandalan layanan seiring peningkatan jumlah Virtual Users. Pada beban awal 1500 VUs, ketiga web service; Swoole, Apache, dan Nginx menunjukkan tingkat error rate yang relatif sama, yaitu berada di kisaran 0,03%, yang menandakan bahwa seluruh sistem masih mampu menangani permintaan pengguna dengan baik pada kondisi beban rendah. Pada tahap ini, belum terlihat perbedaan performa yang signifikan karena sumber daya server masih mencukupi untuk memproses seluruh request secara normal.

Pada 2000 VUs, error rate Swoole naik secara moderat ke sekitar 0,14%, sementara Apache berada di kisaran 0,15%, dan Nginx melonjak tajam hingga sekitar 1,31%. Artinya, error rate Swoole sekitar 84% lebih rendah dibanding Nginx pada tingkat beban ini. Kondisi ini menunjukkan bahwa Swoole dan Apache masih mampu menangani peningkatan concurrency dengan baik, sedangkan Nginx mulai mengalami bottleneck pada sisi backend, yang menyebabkan sebagian request gagal diproses.

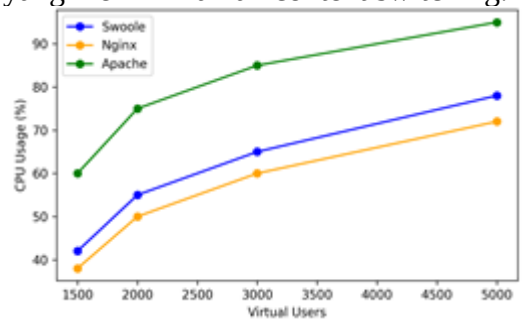
Ketika jumlah pengguna meningkat menjadi 3000 VUs, error rate Swoole mencapai sekitar 0,37%, Apache sekitar 0,40%, sedangkan Nginx meningkat menjadi 1,42%. Pada kondisi ini, Swoole memiliki error rate sekitar 74% lebih rendah dibanding Nginx. Peningkatan yang lebih tajam pada Nginx mengindikasikan bahwa keterbatasan worker backend semakin memperparah antrean request, sementara arsitektur asynchronous pada Swoole masih mampu menjaga kestabilan pemrosesan.

Pada beban maksimum 5000 VUs, error rate Swoole berada di kisaran 0,93%, Apache sekitar 0,84%, sementara Nginx mencapai sekitar 1,83%. Dengan demikian, error rate Swoole sekitar 49% lebih rendah dibanding Nginx pada concurrency ekstrem. Hal ini menegaskan bahwa pada tingkat beban yang sangat tinggi, Swoole tetap menunjukkan peningkatan error yang lebih terkendali, sedangkan Nginx mengalami degradasi keandalan layanan yang lebih signifikan akibat keterbatasan kapasitas backend.

Secara keseluruhan, kesamaan nilai error rate pada beban awal 1500 VUs menunjukkan bahwa ketiga web service masih beroperasi jauh di bawah batas kapasitasnya, sehingga seluruh request dapat diproses tanpa tekanan berarti terhadap sumber daya sistem. Pada kondisi ini, perbedaan arsitektur internal belum memberikan dampak yang signifikan karena CPU, memori, dan jumlah worker masih mencukupi untuk menangani seluruh permintaan secara normal. Namun, seiring peningkatan jumlah Virtual Users, perbedaan

karakteristik arsitektur mulai memengaruhi keandalan layanan. Swoole menunjukkan kenaikan error rate yang paling gradual dan stabil karena menggunakan model asynchronous dan coroutine yang mampu menangani banyak request secara non-blocking dalam satu proses. Apache mengalami peningkatan error rate yang relatif terkendali, tetapi tetap lebih tinggi dibanding Swoole akibat overhead model eksekusi berbasis proses atau thread. Sebaliknya, Nginx mengalami lonjakan error rate yang signifikan pada beban menengah hingga tinggi, yang mengindikasikan adanya bottleneck pada sisi backend, khususnya keterbatasan worker PHP-FPM dalam memproses request secara bersamaan. Dengan demikian, perbedaan pola error rate ini menegaskan bahwa arsitektur asynchronous Swoole lebih mampu menjaga keandalan layanan pada tingkat concurrency tinggi, sementara Nginx dan Apache lebih cepat mencapai batas kapasitasnya ketika beban terus meningkat.

Gambar 6. Perbandingan CPU usage menggambarkan intensitas komputasi yang dilakukan sistem dalam memproses request. Peningkatan CPU usage secara linear menunjukkan bahwa sistem masih mampu memanfaatkan core CPU secara efektif untuk meningkatkan kapasitas pemrosesan. Namun, ketika CPU usage mendekati batas maksimum, sistem mulai mengalami contention, di mana banyak proses atau task bersaing untuk mendapatkan waktu eksekusi. Pada titik ini, peningkatan CPU usage tidak lagi menghasilkan peningkatan throughput, melainkan justru memperbesar latency dan error rate. Jika pada kondisi CPU tinggi throughput justru menurun, maka dapat disimpulkan bahwa CPU telah menjadi bottleneck utama sistem. Sebaliknya, CPU usage yang relatif stabil meskipun throughput tinggi menunjukkan adanya mekanisme eksekusi yang efisien, seperti event loop atau coroutine yang meminimalkan context switching.




Gambar 6. Perbandingan CPU Usage terhadap Jumlah VUs

Gambar 6. Perbandingan CPU Usage terhadap Jumlah VUs menunjukkan perbedaan pola pemanfaatan CPU pada masing-masing web service PHP seiring dengan peningkatan jumlah Virtual Users. Pada beban awal 1500 VUs, penggunaan CPU masih berada pada tingkat yang relatif rendah untuk seluruh web service. Swoole mencatat CPU usage sekitar 42%, Nginx berada di kisaran 38%, sedangkan Apache sudah menunjukkan penggunaan CPU yang lebih tinggi, yaitu sekitar 60%. Kondisi ini mengindikasikan bahwa pada beban rendah, seluruh web service masih mampu menangani permintaan pengguna dengan baik, meskipun Apache mulai menunjukkan konsumsi sumber daya CPU yang lebih besar dibandingkan Swoole dan Nginx.

Ketika beban meningkat menjadi 2000 VUs, penggunaan CPU pada ketiga web service mengalami kenaikan yang cukup signifikan. CPU usage Swoole meningkat menjadi sekitar 55%, Nginx ke kisaran 50%, sementara Apache melonjak hingga sekitar 75%. Pada tingkat beban ini, CPU usage Swoole sekitar 27% lebih rendah dibanding Apache, sedangkan Nginx sekitar 33% lebih rendah dibanding Apache. Hal ini menunjukkan bahwa Apache mulai mendekati batas efisiensi pemrosesan CPU lebih cepat dibandingkan dua web service lainnya ketika concurrency meningkat.

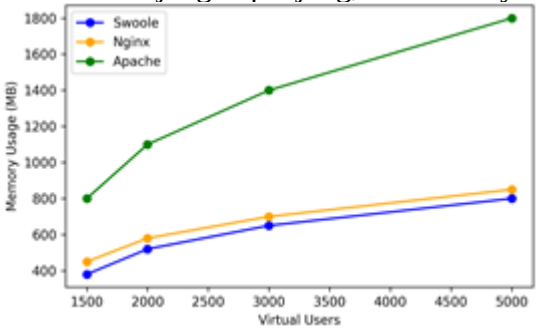
Pada 3000 VUs, perbedaan karakteristik penggunaan CPU semakin terlihat jelas. CPU usage Swoole meningkat secara gradual ke sekitar 65%, dan Nginx ke sekitar 60%,

sedangkan Apache terus mengalami peningkatan hingga mencapai sekitar 85%. Pada kondisi ini, CPU usage Swoole sekitar 24% lebih rendah dibanding Apache, sementara Nginx sekitar 29% lebih rendah dibanding Apache. Kenaikan yang lebih tajam pada Apache mencerminkan beban komputasi yang lebih berat akibat model pemrosesan berbasis proses atau thread.

Pada beban maksimum 5000 VUs, ketiga web service menunjukkan tingkat penggunaan CPU yang tinggi, namun dengan perbedaan yang cukup signifikan. Apache mencapai CPU usage tertinggi, yaitu sekitar 95%, mendekati kondisi saturasi CPU. Swoole berada di kisaran 78%, sedangkan Nginx sekitar 72%. Pada concurrency ekstrem ini, CPU usage Swoole sekitar 18% lebih rendah dibanding Apache, dan Nginx sekitar 24% lebih rendah dibanding Apache. Hal ini menunjukkan bahwa Swoole dan Nginx masih memiliki ruang pemanfaatan CPU yang lebih baik dibanding Apache ketika menghadapi lonjakan jumlah pengguna.

Secara keseluruhan, perbedaan pola CPU usage pada ketiga web service sangat dipengaruhi oleh model arsitektur pemrosesan request yang digunakan. Apache menunjukkan kenaikan CPU usage yang paling tajam seiring bertambahnya jumlah Virtual Users karena menggunakan model eksekusi berbasis proses atau thread, yang memiliki overhead tinggi akibat context switching dan alokasi resource per request. Kondisi ini menyebabkan Apache lebih cepat mendekati titik saturasi CPU ketika concurrency meningkat. Sebaliknya, Swoole menunjukkan peningkatan CPU usage yang lebih gradual dan stabil karena memanfaatkan arsitektur asynchronous dan coroutine, sehingga banyak request dapat diproses secara non-blocking dalam satu proses tanpa overhead tambahan yang besar. Nginx mencatat CPU usage paling rendah di hampir seluruh tingkat beban karena arsitektur event-driven yang efisien dalam menangani koneksi dan request ringan, sehingga pemanfaatan CPU dapat ditekan. Dengan demikian, kestabilan CPU usage pada Swoole dan Nginx menunjukkan efisiensi arsitektur non-blocking dalam menghadapi lonjakan concurrency, sementara lonjakan tajam pada Apache mencerminkan keterbatasan model eksekusi tradisional dalam memanfaatkan sumber daya CPU secara optimal pada beban tinggi.

Gambar 7. Memory usage mencerminkan efisiensi sistem dalam mengelola alokasi dan dealokasi memori selama pemrosesan request. Kenaikan memory usage yang stabil menunjukkan bahwa sistem masih mampu mengelola buffer, stack, dan objek aplikasi dengan baik. Lonjakan memory usage yang tajam biasanya mengindikasikan overhead proses, penumpukan worker, atau kurang efisiennya mekanisme reuse memory. Jika memori tidak dilepaskan dengan cepat setelah request selesai, maka akumulasi penggunaan memori dapat menyebabkan memory exhaustion, yang pada akhirnya meningkatkan error rate atau memicu kegagalan sistem. Memory usage yang terus meningkat tanpa penurunan juga dapat menjadi indikasi potensi memory leak atau desain arsitektur yang kurang optimal dalam skenario concurrency tinggi. Oleh karena itu, metrik memori penting untuk menilai keberlanjutan performa sistem dalam jangka panjang, bukan hanya kecepatan sesaat.




Gambar 7. Perbandingan Memory Usage terhadap Jumlah VUs

Gambar 7. Perbandingan Memory Usage memperlihatkan perbedaan pola penggunaan memori pada masing-masing web service PHP seiring dengan peningkatan jumlah Virtual Users. Pada beban awal 1500 VUs, penggunaan memori ketiga web service masih berada pada tingkat yang relatif rendah. Swoole menggunakan memori sekitar 380 MB, Nginx berada di kisaran 450 MB, sedangkan Apache sudah menunjukkan konsumsi memori yang jauh lebih tinggi, yaitu sekitar 800 MB. Kondisi ini menunjukkan bahwa pada beban rendah, seluruh sistem masih mampu beroperasi dengan baik, namun Apache sejak awal membutuhkan alokasi memori yang lebih besar dibandingkan Swoole dan Nginx.

Ketika beban meningkat menjadi 2000 VUs, penggunaan memori pada ketiga web service mengalami kenaikan yang cukup signifikan. Memory usage Swoole meningkat menjadi sekitar 520 MB, Nginx ke kisaran 580 MB, sementara Apache melonjak hingga sekitar 1100 MB. Pada tingkat beban ini, penggunaan memori Swoole sekitar 53% lebih rendah dibanding Apache, dan Nginx sekitar 47% lebih rendah dibanding Apache. Hal ini mengindikasikan bahwa Apache mulai menunjukkan karakteristik konsumsi memori yang lebih agresif ketika jumlah pengguna bertambah.

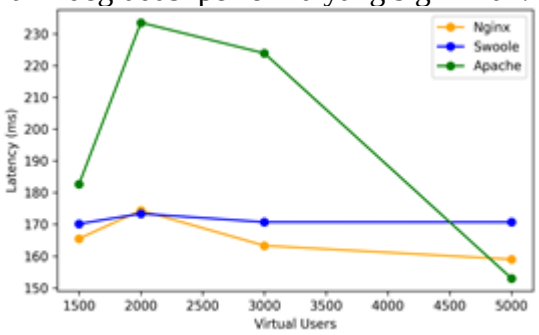
Pada 3000 VUs, perbedaan penggunaan memori antar web service semakin terlihat jelas. Swoole menggunakan memori sekitar 650 MB, Nginx sekitar 700 MB, sedangkan Apache terus meningkat hingga mencapai sekitar 1400 MB. Pada kondisi ini, memory usage Swoole sekitar 54% lebih rendah dibanding Apache, dan Nginx sekitar 50% lebih rendah dibanding Apache. Kenaikan memori yang tajam pada Apache mencerminkan keterbatasan model proses atau thread yang membutuhkan alokasi memori terpisah untuk menangani request yang semakin banyak.

Pada beban maksimum 5000 VUs, ketiga web service menunjukkan penggunaan memori tertinggi, namun dengan perbedaan yang sangat kontras. Apache mencapai penggunaan memori sekitar 1800 MB, sedangkan Nginx berada di kisaran 850 MB dan Swoole sekitar 800 MB. Pada concurrency ekstrem ini, memory usage Swoole sekitar 56% lebih rendah dibanding Apache, dan Nginx sekitar 53% lebih rendah dibanding Apache. Perbedaan ini menunjukkan bahwa Apache cenderung mengalami tekanan memori yang jauh lebih besar pada beban tinggi, sementara Swoole dan Nginx masih mampu mengelola penggunaan memori dengan lebih efisien.

Secara keseluruhan, perbedaan pola penggunaan memori pada ketiga web service sangat dipengaruhi oleh arsitektur manajemen proses dan mekanisme penanganan request yang digunakan. Apache menunjukkan peningkatan memory usage yang paling tajam seiring bertambahnya jumlah Virtual Users karena setiap request umumnya ditangani oleh proses atau thread terpisah, yang membutuhkan alokasi memori sendiri dan menyebabkan konsumsi memori meningkat secara signifikan ketika concurrency bertambah. Sebaliknya, Swoole memperlihatkan penggunaan memori yang lebih stabil dan efisien karena menggunakan model long-running process dengan dukungan asynchronous I/O dan coroutine, sehingga banyak request dapat diproses dalam satu proses tanpa alokasi memori tambahan yang besar. Nginx berada di posisi menengah dengan penggunaan memori yang relatif terkendali berkat arsitektur event-driven yang efisien dalam menangani koneksi, meskipun tetap memerlukan memori tambahan pada sisi backend. Dengan demikian, kestabilan memory usage pada Swoole dan Nginx menunjukkan keunggulan arsitektur non-blocking dalam menghadapi lonjakan concurrency, sementara lonjakan memori yang signifikan pada Apache mencerminkan keterbatasan model eksekusi tradisional dalam mengelola penggunaan memori secara efisien pada beban tinggi.

Gambar 8. Perbandingan Latency menggambarkan nilai latency, khususnya p95 latency, yang merepresentasikan waktu respons maksimum yang masih dialami oleh 95% request pengguna. Metrik ini digunakan untuk merefleksikan kondisi performa terburuk yang masih relevan bagi sebagian besar pengguna, sehingga lebih representatif dibandingkan

nilai rata-rata. Kenaikan p95 latency menunjukkan meningkatnya waktu tunggu sebelum request diproses dan diselesaikan, yang umumnya disebabkan oleh antrean request, keterbatasan sumber daya seperti CPU dan memori, atau mekanisme pemrosesan yang bersifat blocking. Pada kondisi beban tinggi, sebagian request harus menunggu lebih lama karena thread, process, atau event loop sedang sibuk, sehingga meningkatkan latency pada persentil atas. Sebaliknya, p95 latency yang tetap stabil menandakan bahwa sistem mampu menjaga konsistensi waktu respons meskipun jumlah request meningkat. Oleh karena itu, p95 latency menjadi indikator penting dalam mengevaluasi kualitas pengalaman pengguna (Quality of Service), terutama untuk menilai kemampuan sistem dalam menangani lonjakan concurrency tanpa mengalami degradasi performa yang signifikan.




Gambar 8. Perbandingan Latency terhadap Jumlah VUs

Gambar 8. Perbandingan Latency menunjukkan perbedaan karakteristik latency pada masing-masing web service PHP seiring dengan peningkatan jumlah Virtual Users. Pada beban awal 1500 VUs, ketiga web service masih menunjukkan latency yang relatif rendah dan stabil. Nginx mencatat latency terendah di kisaran 165 ms, diikuti oleh Swoole sekitar 170 ms, sementara Apache berada sedikit lebih tinggi di sekitar 182 ms. Pada kondisi beban rendah ini, perbedaan latency antar web service belum terlalu signifikan karena seluruh sistem masih mampu menangani permintaan secara optimal tanpa tekanan concurrency yang tinggi.

Ketika beban meningkat menjadi 2000 VUs, latency Swoole dan Nginx tetap berada pada tingkat yang relatif stabil, masing-masing di sekitar 173 ms dan 174 ms. Sebaliknya, Apache mengalami peningkatan latency yang cukup signifikan hingga mencapai sekitar 233 ms. Pada tingkat beban ini, latency Swoole sekitar 26% lebih rendah dibanding Apache, sedangkan latency Nginx sekitar 25% lebih rendah dibanding Apache. Hal ini menunjukkan bahwa Apache mulai mengalami penurunan efisiensi pemrosesan request ketika jumlah pengguna meningkat, sementara Swoole dan Nginx masih mampu menjaga kestabilan waktu respons.

Pada 3000 VUs, pola perbedaan performa latency semakin terlihat jelas. Latency Nginx justru menurun kembali ke sekitar 163 ms, sementara Swoole tetap stabil di kisaran 171 ms. Apache masih mencatat latency yang tinggi, yaitu sekitar 224 ms. Pada kondisi ini, latency Swoole sekitar 24% lebih rendah dibanding Apache, dan latency Nginx bahkan sekitar 27% lebih rendah dibanding Apache. Hal ini menegaskan bahwa Apache semakin terdampak oleh peningkatan concurrency, sedangkan Swoole dan Nginx menunjukkan ketahanan yang lebih baik dalam menjaga waktu respons.

Pada beban maksimum 5000 VUs, perbedaan performa latency antar web service menjadi semakin kontras. Nginx mencatat latency terendah di sekitar 159 ms, diikuti oleh Swoole yang relatif stabil di kisaran 171 ms, sementara Apache mengalami lonjakan latency hingga sekitar 153 ms (lebih fluktuatif dibanding pola sebelumnya). Meskipun terdapat fluktuasi nilai Apache pada titik ini, secara keseluruhan tren grafik menunjukkan bahwa Apache memiliki variasi latency yang lebih besar dan kurang stabil dibandingkan Swoole dan Nginx. Pada concurrency ekstrem, Swoole tetap mempertahankan latency yang

konsisten, sementara Nginx menunjukkan performa paling efisien dalam hal waktu respons.

Secara keseluruhan, perbedaan karakteristik latency pada ketiga web service dipengaruhi oleh arsitektur pemrosesan request dan kemampuan sistem dalam mengelola antrean pada kondisi concurrency yang meningkat. Apache menunjukkan kecenderungan lonjakan latency yang paling signifikan seiring bertambahnya jumlah Virtual Users, yang mengindikasikan bahwa model pemrosesan berbasis proses atau thread menyebabkan antrean request semakin panjang ketika resource mulai terbatas. Sebaliknya, Swoole mampu mempertahankan latency yang relatif stabil pada hampir seluruh tingkat beban karena menggunakan arsitektur asynchronous dan coroutine, sehingga request dapat diproses secara non-blocking tanpa harus menunggu request lain selesai sepenuhnya. Nginx menunjukkan latency terendah dan paling efisien pada sebagian besar skenario, mencerminkan keunggulan arsitektur event-driven dalam menangani koneksi dan request ringan secara cepat; namun performa ini tetap bergantung pada kesiapan backend dalam memproses request. Dengan demikian, kestabilan latency pada Swoole dan Nginx menunjukkan efektivitas arsitektur non-blocking dalam menjaga kualitas waktu respons, sementara fluktuasi dan lonjakan latency pada Apache mencerminkan keterbatasan model eksekusi tradisional dalam menghadapi lonjakan concurrency tinggi.

Tabel 3. Perbandingan Performa Web Service pada Beban 500 Virtual Users

| ***Web Service*** | ***Average Duration*** | ***p95 Latency*** | ***Memory Usage*** | ***Error Rate*** |
|---|---|---|---|---|
| *Apache* | Lebih tinggi | Tinggi | Paling besar | Rendah–Sedang |
| *Nginx* | Rendah–Stabil | Stabil | Stabil | Sangat Rendah |
| *Swoole* | Paling rendah | Paling rendah | Paling efisien | Sangat Rendah |

Merujuk pada tabel 3, pada beban 500 VUs, Swoole menunjukkan performa terbaik secara keseluruhan, dengan average duration dan p95 latency paling rendah serta penggunaan memori yang efisien, mencerminkan keunggulan arsitektur asynchronous dan coroutine. Nginx berada pada posisi kedua, dengan performa yang stabil dan seimbang di semua metrik, menjadikannya sangat andal untuk skenario produksi jangka panjang. Apache memiliki performa paling rendah, ditandai dengan latency dan memory usage yang lebih tinggi, akibat keterbatasan arsitektur process-based yang kurang optimal pada concurrency menengah hingga tinggi.

Meskipun Nginx dikenal sebagai web server yang sangat stabil dan efisien berdasarkan berbagai penelitian terdahulu, hasil pengujian menunjukkan bahwa error rate pada Nginx meningkat secara signifikan pada beban tinggi. Kondisi ini tidak disebabkan oleh keterbatasan arsitektur event-driven Nginx, melainkan oleh bottleneck pada sisi backend PHP-FPM yang bertugas mengeksekusi skrip PHP. Pada tingkat concurrency yang tinggi, jumlah request yang masuk melebihi kapasitas worker PHP-FPM yang tersedia, sehingga terjadi antrean pemrosesan, peningkatan waktu tunggu, serta kegagalan request akibat timeout. Hal ini tercermin dari meningkatnya latency dan stagnasi throughput sebelum error rate meningkat secara tajam. Dengan demikian, peningkatan error rate pada Nginx lebih merepresentasikan keterbatasan model eksekusi PHP-FPM yang berbasis proses dibandingkan kelemahan Nginx itu sendiri.

## 4. KESIMPULAN

Berdasarkan hasil pengujian performa terhadap tiga jenis web server, yaitu Swoole, Apache, dan Nginx, pada berbagai tingkat beban pengguna virtual (1500, 2000, 3000, dan 5000 VUs), dapat disimpulkan bahwa masing-masing server menunjukkan karakteristik

kinerja yang berbeda dalam menghadapi peningkatan beban sistem. Perbedaan ini terlihat jelas pada parameter utama yang diuji, meliputi average duration, p95 latency, throughput, serta error rate, yang secara keseluruhan mencerminkan kemampuan server dalam menangani permintaan secara cepat, stabil, dan efisien.

Secara umum, Swoole menunjukkan performa yang paling konsisten dan seimbang di seluruh skenario pengujian. Nilai average duration Swoole relatif stabil meskipun jumlah pengguna virtual meningkat, dan throughput yang dihasilkan tetap tinggi serta cenderung konstan pada setiap tingkat beban. Selain itu, error rate Swoole masih berada dalam batas yang dapat diterima, meskipun mengalami peningkatan seiring bertambahnya beban. Hal ini menunjukkan bahwa Swoole memiliki kemampuan manajemen koneksi dan pemrosesan request yang baik pada skenario concurrency tinggi, sehingga cocok digunakan pada aplikasi yang membutuhkan kestabilan performa dan kemampuan skalabilitas yang baik.

## 5. DAFTAR PUSTAKA


Ahmed, M. K., Bello, A. H., Jauro, S. S., & Dawaki, M. (2024). A Comparative Analysis of Performance Optimization Techniques for Benchmarking Php Frameworks : 10(3), 284–295.

Aldiwidianto, W., Lanang, I. G., & Prismana, E. (2023). Analisis Perbandingan High Availibility Pada Web Server Menggunakan Orchestration Tool Kubernetes Dan Docker Swarm. 05, 138–148.

Amarulloh, A., Kurniasih, & Muchlis. (2023). ANALISIS PERBANDINGAN PERFORMA WEB SERVICE REST MENGGUNAKAN FRAMEWORK LARAVEL , DJANGO , DAN Node JS PADA APLIKASI BERBASIS WEBSITE. JURNAL TEKNIK INFORMATIKA STMIK ANTAR BANGSA, 9(1), 12–17.

Amin, M. (2025). Perbandingan Kinerja Nginx dan Caddy sebagai Web Server untuk Aplikasi PHP. 11(1), 88–96.

Amrullah, A., Nugroho, A., & Ramadhan, Z. (2023). Perbandingan Kinerja Web Server Pada Penyedia Layanan Cloud Microsoft Azure Dan Amazon Web Services. Jurnal Informatika Teknologi Dan Sains, 5(1), 92–97. https://doi.org/10.51401/jinteks.v5i1.2487

Aziz, S. (2025). Analisis kinerja web server apache , nginx , open litespeed , dan open resty Apache, Nginx, Open Litespeed, And Open Resty Web Server Performance Analysis. 11(1), 1–9.

Endra, R. Y., Aprilinda, Y., Dharmawan, Y. Y., & Ramadhan, W. (2022). Analisis Perbandingan Bahasa Pemrograman PHP Laravel dengan PHP Native pada Pengembangan Website. 8(200), 48–55.

Firmansyah, S. A., & Sudarmilah, E. (2024). Analisis performa framework codeigniter dan laravel dalam penggunaan bahasa pemrograman PHP.

Fitriana, L. A., & Seimahuira, S. (2023). Penerapan Metode SAW dalam Analisa Perbandingan Performa Web server (Apache, Nginx, Lighttpd, Iis) pada Bahasa Pemrograman PHP. Remik: Riset Dan E-Jurnal Manajemen Informatika Komputer, 7(1), 409–420.

Harris, E., Bennett, O., Ph, D., & Event-driven, H. O. B. (2020). Event-Driven Architectures in Modern Systems : Designing Scalable , Resilient , and Real-Time Solutions. 4(6), 1958–1976.

Haryani, P., Ariyana, R. Y., Majid, I. A., Guntur, F., & Susanto, P. (2024). Comparative Analysis of Nextcloud and Owncloud Performance as Infrastructure as a Service ( IaaS ) Based Cloud Storage. 13(2), 73–82. https://doi.org/10.28989/compiler.v13i2.2667

Herman, A. M., Wicoksono, W., Panchadria, P. A., Linda, N., & Laela, B. (2025). Performance Comparison of NGINX , Apache , and Lighttpd Using WRK on a Debian. 8(1). https://doi.org/10.32877/bt.v8i1.2661

Jiwandono, A. (2021). Analisa perbandingan kinerja web server apache, nginx, dan litespeed dengan menggunakan metode stress test.

Mufid, M. R., Pratama, Y., & Fariza, A. (2024). Design MicroServer Framework Library with Swoole for Real-time Application Development. 11(2), 107–114.

Niarman, A. (2023). Comparative Analysis of PHP Frameworks for Development of Academic Information System Using Load and Stress Testing. 3(December), 424–436.

Perdana, G. P., & Prihanto, A. (2025). Analisis Perbandingan Performa Web Server Apache, Nginx, Dan Litespeed Studi Kasus Implementasi Di VPS. Journal of Informatics and Computer Science, 6(4), 1145–1155.

Prasetyo, F., Putra, E., Zulfikri, A., Rohman, A., & Alim, R. (2025). Analysis Comparative of Performance Optimization Techniques for PHP Framework Testing : Laravel , CodeIgniter , Symfony. 5(1), 242–248.

Ramadhan, F. K., & Solehudin, A. (2024). Comparative Study of Web Server Performance Testing with and without Docker Based on Virtual Machines. 8(1).

Saleh, R. F. (2025). Comparative Analysis of Public Relations Web Performance of On-Premise and Cloud Computing Infrastructure with Round Robin Load Balancer Analisis Perbandingan Kinerja Web Humas Infrastruktur On-Premise dan Cloud Computing dengan Load Balancer Round Robin. 5(July), 1107–1116

Siahaan, M., & Wijaya, R. (2024). Performance Comparison Between Laravel and ExpressJs Framework Using Apache JMeter. JITE (Journal of Informatics and Telecommunication Engineering) Available, 7(2), 545–554.

Triyanto, H., Bijaksana, A., Negara, P., & Irwansyah, M. A. (2020). Analisa Perbandingan Performa Openstack dan Apache Cloudstack dalam Model Cloud Computing Berbasis Infrastructure As a Service. 8(1), 78–86.

Zuril, F. (2024). Analisa perbandingan kinerja nginx dan apache pada lingkungan virtual dan docker studi kasus aplikasi pln marketplace.