Analisis Komparatif Performa Apache2 dan Nginx pada Aplikasi CRUD Berbasis Ubuntu Server

Arwani[1], Ahmad Tahqiq Rofiq[2], Bagas Ardiansyah Pratama[3], Abdus Samad[4]
[1,2,3,4] Teknologi Informasi, Universitas Ibrahimy, Situbondo
Email : [*1]arwaninew523@gmail.com, [2] ahmadtahqiq@gmail.com, begekdnt@gmail.com, [4]saintek.somad@gmail.com

Abstrak

Perkembangan aplikasi web dinamis menuntut infrastruktur server yang responsif dan efisien. Masalah utama yang sering dihadapi adalah penurunan kinerja server saat menangani lonjakan trafik simultan pada operasi database dinamis. Penelitian ini bertujuan menganalisis secara komparatif efisiensi kinerja antara web server Apache2 dan Nginx dalam mengelola aplikasi CRUD berbasis PHP dan MySQL pada lingkungan Ubuntu Server di VirtualBox yang dijalankan di atas komputer host berspesifikasi Processor Intel Core i5 dan RAM 16 GB. Metode penelitian menggunakan pendekatan eksperimen laboratorium melalui stress testing menggunakan perkakas Apache Benchmark (ab) dengan parameter 1000 total permintaan dan 100 koneksi simultan, serta pengawasan sumber daya menggunakan utilitas htop. Parameter evaluasi yang diukur meliputi Requests per Second (RPS), latensi, dan efisiensi konsumsi memori (RAM). Hasil penelitian menunjukkan bahwa Apache2 memiliki throughput sedikit lebih tinggi dengan capaian 225,13 RPS, unggul 8,48% dibandingkan Nginx (207,53 RPS). Namun, dalam aspek stabilitas respons pada trafik puncak, Nginx jauh lebih andal dengan latensi maksimal hanya 609 ms, sedangkan Apache2 melonjak ekstrem hingga 1269 ms. Selain itu, Nginx terbukti jauh lebih efisien dalam manajemen memori dengan hanya mengalami lonjakan bersih RAM sebesar 67 MB saat beban puncak, dibandingkan Apache2 yang menghabiskan hingga 249 MB. Kesimpulannya, Apache2 efektif untuk throughput rata-rata beban menengah, sementara Nginx unggul dalam menjaga konsistensi waktu respons serta efisiensi sumber daya pada beban kerja tinggi di lingkungan virtual terbatas.

Kata Kunci: *Apache2, Nginx, Web Server, Benchmarking, CRUD*

PENDAHULUAN

Pesatnya transformasi digital menempatkan aplikasi web sebagai pilar utama di berbagai sektor. Seiring melonjaknya pengguna internet, infrastruktur server dituntut mampu melayani permintaan (*request*) bervolume besar secara simultan. Masalah utama yang sering dihadapi administrator sistem adalah terjadinya degradasi performa atau kegagalan server (*crash*) saat aplikasi menerima lonjakan trafik mendadak.(Azi et al., 2023). Hal ini dipicu oleh ketidakmampuan arsitektur *web server* dalam mengelola antrean koneksi simultan (*high concurrency*) secara efisien saat mengeksekusi skrip sisi server (*server-side scripting*) dan interaksi basis data.(Faisal et al., 2025).

Untuk mengatasinya, diperlukan konfigurasi *web server* yang optimal sesuai karakteristik beban kerja. Dua teknologi yang mendominasi pasar saat ini adalah Apache dan Nginx. Apache menggunakan pendekatan *Process-Based* yang rentan memicu *overhead* memori pada trafik padat. Sebaliknya, Nginx menerapkan paradigma *Event-Driven* dan *Asynchronous* yang mampu menangani ribuan koneksi simultan dengan memori minimal. Pengujian komparatif diperlukan untuk memetakan performa objektif kedua server tersebut.(Saputra et al., 2023)

Beberapa penelitian lima tahun terakhir menunjukkan Nginx unggul pada konten statis dan lebih adaptif di lingkungan terbatas. Namun, mayoritas studi terdahulu berfokus pada konten statis atau

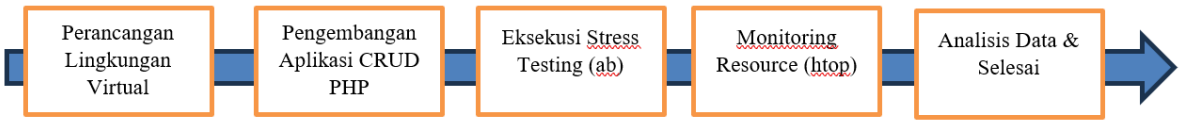
infrastruktur skala besar. Sangat sedikit penelitian yang membedah performa pada lingkungan virtualisasi bersumber daya terbatas (seperti VirtualBox) untuk mengeksekusi operasi dinamis CRUD (*Create, Read, Update, Delete*). Operasi CRUD memiliki kompleksitas tinggi karena menguji komunikasi *real-time* antara *engine* PHP dan basis data MySQL, yang menjadi nilai kebaruan (*novelty*) penelitian ini.

Penelitian ini bertujuan menganalisis secara komparatif kinerja Apache2 dan Nginx pada virtual Ubuntu Server. Parameter yang dievaluasi meliputi *Requests per Second* (RPS), *Mean Latency*, distribusi latensi, dan efisiensi RAM, guna memberikan rekomendasi teknis berbasis data empiris. Namun, perlu digarisbawahi bahwa ruang lingkup hasil pengujian dalam penelitian ini terikat pada batasan spesifikasi lingkungan virtualisasi VirtualBox tertentu, sehingga karakteristik performa yang ditemukan tidak dapat langsung digeneralisasikan secara mutlak pada seluruh kasus lingkungan server produksi nyata.

METODE

A. Tahapan Penelitian

Penelitian ini dijalankan melalui pendekatan eksperimen laboratorium terstruktur guna menjamin akurasi dan konsistensi data performa web server yang diperoleh. Rangkaian sistematis dari awal hingga akhir eksperimen diadopsi untuk memastikan bahwa seluruh parameter dapat diukur secara objektif dan dapat diuji kembali (repeatable).(Nurazizah et al., 2024) Alur kerja dari pelaksanaan penelitian ini secara menyeluruh dituangkan pada Gambar 1.



Gambar 1 Alur kerja pelaksanaan penelitian

Berdasarkan Gambar 1, penjelasan dari masing-masing tahapan alur penelitian adalah sebagai berikut:

1. Perancangan Lingkungan Virtual: Tahap awal melibatkan instalasi dan konfigurasi perangkat lunak sistem operasi Guest di dalam platform virtualisasi untuk menciptakan lingkungan server yang terisolasi.
2. Pengembangan Aplikasi CRUD PHP: Pembuatan kode program aplikasi berbasis web dinamis beserta struktur basis datanya untuk digunakan sebagai objek pengujian beban kerja koneksi.
3. Eksekusi Stress Testing: Proses pemberian beban kerja simultan menggunakan perkakas pengujian dari mesin Host ke mesin Guest secara bergantian pada masing-masing web server.
4. Monitoring Resource: Mengamati dan mencatat perilaku penggunaan memori utama dan prosesor pada server secara real-time menggunakan utilitas internal saat beban puncak terjadi.
5. Analisis Data: Analisis Data: Data empiris berupa angka statistik *throughput* dan latensi dikumpulkan untuk dihitung persentase selisihnya dan ditarik kesimpulan ilmiah.

B. Spesifikasi Lingkungan Eksperimen dan Formulasi Pengujian

Metode penyelesaian masalah dalam penelitian ini didasarkan pada pengujian komparatif Out-of-the-Box pada berkas konfigurasi bawaan masing-masing vendor perangkat lunak server.(Zulfahrizan et al., 2025). Untuk menjaga keadilan pengujian, spesifikasi lingkungan virtual disamakan sepenuhnya.(Rafli et al., 2022) Konfigurasi perangkat lunak sistem manajemen basis data yang digunakan sebagai komponen krusial dalam arsitektur aplikasi CRUD dinamis ini dijabarkan pada Tabel 1.

Tabel 1. Spesifikasi Komponen Basis Data Eksperimen

Nama Komponen Sistem	Versi Perangkat Lunak	Port Komunikasi	Peran Eksperimen
Ubuntu Server	22.04 LTS	22 (SSH)	Sistem Operasi Guest (VM)
Apache HTTP Server	2.4.52	80 (HTTP)	Web Server Uji 1 (Process-Based)
Nginx Engine-X	1.18.0	80 (HTTP)	Web Server Uji 2 (Event-Driven)
PHP Engine	8.1.2	Unix Socket	Interpreter Skrip CRUD PHP
MySQL Database	8.0	3306	Sistem Manajemen Basis Data

Rincian konfigurasi *out-of-the-box* pada kedua *web server* diatur untuk berjalan pada mode standar masing-masing vendor tanpa adanya optimasi manual.(Handayani et al., 2026) Pada Apache HTTP Server v2.4.52, arsitektur dijalankan menggunakan *Multi-Processing Module* (MPM) jenis *Event* secara *default*. Untuk Nginx Engine-X v1.18.0, konfigurasi diatur menggunakan 1 *worker process* dengan batasan *worker connections* sebesar 768 koneksi. Sementara itu, PHP-FPM v8.1.2 dikonfigurasi menggunakan proses manajer dinamis (*dynamic process manager*) dengan parameter bawaan pm.max_children = 5, pm.start_servers = 2, pm.min_spare_servers = 1, dan pm.max_spare_servers = 3, yang saling berkomunikasi dengan server utama melalui *Unix Domain Socket*.

Penjelasan dari Tabel 1 menunjukkan bahwa seluruh arsitektur sistem berjalan lokal di dalam satu mesin virtual tunggal. Komunikasi antara skrip PHP dengan database MySQL diintegrasikan melalui ekstensi MySQLi pada port standar 3306. Skenario beban kerja diatur menggunakan parameter formulasi matematis untuk menentukan total kapasitas kerja yang masuk ke sistem. Hubungan antara tingkat kepadatan koneksi simultan, total permintaan, dan total waktu pemrosesan secara teoritis mengikuti persamaan kalkulasi *throughput* standar sebagai berikut:

$$\text{Requests per Second (RPS)} = \frac{N}{T} \quad (1)$$

Di mana N merupakan total requests yang dikirimkan oleh perkakas uji (dalam penelitian ini diatur sebanyak 1000 permintaan), dan T merupakan total waktu *(time taken for tests)* dalam satuan detik yang dibutuhkan oleh *web server* untuk menyelesaikan seluruh antrean permintaan tersebut. (Lohja et al., 2023) Sementara itu, untuk menghitung tingkat efisiensi selisih performa dalam bentuk persentase kuantitatif antara kedua objek uji, digunakan rumus sebagai berikut:

$$\Delta P = \frac{(|P_{apache} - P_{nginx}|)}{P_{nginx}} \times 100\% \quad (2)$$

Di mana ΔP merepresentasikan nilai persentase selisih kinerja, sedangkan P_{apache} dan P_{nginx} masing-masing adalah parameter angka performa murni yang dihasilkan oleh Apache2 dan Nginx selama proses pengujian beban berlangsung.

HASIL DAN PEMBAHASAN

Bagian ini menyajikan data empiris hasil *stress testing* serta analisis komparatif performa antara *web server* Apache2 dan Nginx pada lingkungan Ubuntu Server. Pengujian difokuskan pada replikasi beban kerja dinamis aplikasi CRUD berbasis PHP dan MySQL menggunakan alokasi sumber

daya virtual yang terbatas. Melalui hasil pengujian ini, karakteristik arsitektur kedua *web server* akan dibedah secara mendalam, baik dari aspek efisiensi pemrosesan jaringan (*throughput* dan latensi) maupun optimalisasi konsumsi sumber daya perangkat keras (CPU dan RAM).(Maila et al., 2020)

A. Hasil Pengujian Kinerja Jaringan

Pengujian performa dilakukan secara bergantian pada masing-masing web server di dalam mesin virtual Ubuntu Server 22.04 LTS.(Warman & Nugraha, 2017) Seluruh data kuantitatif diperoleh secara objektif menggunakan instrumen pengujian Apache Benchmark (ab) yang dijalankan dari komputer Host dengan parameter uji berupa beban kerja dinamis dari halaman indeks aplikasi CRUD yang mengeksekusi query ke database MySQL.

a. Analisis Kuantitas Kapasitas Transaksi (*Throughput*)

Throughput diukur berdasarkan kemampuan dari masing-masing perangkat lunak server dalam merespon dan menyelesaikan seluruh paket permintaan masuk dalam satuan waktu satu detik. Rekaman statistik utama yang dihasilkan oleh perkakas uji dari skenario 1000 total permintaan dengan tingkat kepadatan 100 koneksi bersamaan dijabarkan secara terperinci pada Tabel 2.

Tabel 2. Statistik Performa Utama Hasil Pengujian Beban

Parameter Indikator Performa	Nginx 1.18.0 (PHP-FPM)	Apache 2.4.52	Selisih Efisiensi (%)
Requests per Second (RPS)	207.53	225.13	8.48% (Apache)
Time per Request (Mean)	481.856 ms	444.183 ms	7.82% (Apache)
Transfer Rate (KB/sec)	331.16	369.36	11.53% (Apache)
Failed Requests (Total)	0	0	0.00%
Total Time for Tests	4.819 detik	4.442 detik	7.82% (Apache)

Penjelasan analisis dari rekaman data kuantitatif pada Tabel 2 dapat dijabarkan melalui poin-poin evaluasi di bawah ini:

1. Kapasitas *Requests per Second* (RPS) yang dihasilkan oleh Apache2 mampu menyentuh angka 225.13 RPS, yang berarti arsitektur server tersebut dapat memproses 225 transaksi HTTP dalam satu detik. Di sisi lain, Nginx berada sedikit di bawahnya dengan raihan 207.53 RPS. Melalui perhitungan matematis menggunakan rumus persentase selisih, ditemukan bahwa Apache2 memiliki keunggulan kuantitas pemrosesan data sebesar 8.48% dibandingkan dengan Nginx.
2. Parameter *Time per Request (Mean)* memperlihatkan rerata waktu tunggu yang dialami oleh seluruh total antrean koneksi klien. Apache2 mencatatkan durasi respon rata-rata yang lebih cepat yaitu sebesar 444.183 ms, sedangkan Nginx membutuhkan waktu sebesar 481.856 ms. Selisih durasi pemrosesan halaman CRUD dinamis ini menunjukkan angka efisiensi sebesar 7.82% yang menguntungkan model pemrosesan internal milik Apache2.
3. Nilai *Transfer Rate* mencerminkan kemampuan lebar pita jalur data dalam menggelontorkan muatan dokumen dari memori RAM server menuju ke sisi klien. Apache2 mampu menyalurkan data sebesar 369.36 KB per detik, atau lebih tinggi 11.53% daripada Nginx yang berada pada kecepatan transfer data sebesar 331.16 KB per detik.
4. Nilai *Failed Requests* pada kedua server menunjukkan angka mutlak 0 permintaan yang gagal. Data ini membuktikan bahwa meskipun lingkungan virtual komputer diberikan tekanan beban koneksi simultan yang cukup padat, tidak ada satu pun paket instruksi aplikasi CRUD PHP yang mengalami kegagalan kirim atau pemutusan koneksi sepihak oleh sistem.

b. Distribusi Latensi dan Tingkat Konsistensi Layanan

Meskipun pengujian rata-rata menempatkan Apache2 sebagai server yang mengeksekusi kode program secara lebih cepat, pengamatan mendalam terhadap sebaran waktu respon memberikan indikasi yang berbeda terkait tingkat konsistensi layanan.(Luthfi et al., 2018). Rekaman distribusi latensi yang terbagi berdasarkan persentase volume antrean transaksi disajikan pada Tabel 3.

Tabel 3. Sebaran Distribusi Latensi Berdasarkan Volume Permintaan

Persentase Antrean Permintaan (%)	Latensi Respon Nginx (ms)	Latensi Respon Apache2 (ms)
50% (*Median Response*)	460	346
66%	483	378
75%	499	402
80%	511	425
90%	545	1021
95%	568	1106
98%	585	1125
99%	594	1137
100% (*Longest Request*)	609	1269

Penjelasan dari pola sebaran data distribusi latensi pada Tabel 3 dapat diuraikan secara logis melalui poin-poin evaluasi berikut:

1. Pada kondisi pemrosesan awal hingga beban menengah yang diwakili oleh rentang volume 50% sampai 80% dari total antrean, Apache2 konsisten mempertahankan performa kecepatan respon yang sangat tinggi di kisaran 346 ms hingga 425 ms, mengungguli Nginx yang berjalan konstan di angka 460 ms hingga 511 ms.
2. Titik balik perilaku performa terjadi ketika volume antrean memasuki fase kritis beban puncak, yaitu pada volume permintaan 90%. Pada titik tersebut, latensi Apache2 melonjak drastis secara eksponensial melampaui ambang batas psikologis satu detik hingga menyentuh angka 1021 ms. Sementara itu, Nginx mampu mempertahankan stabilitas kurva latensi dengan hanya mengalami peningkatan tipis ke angka 545 ms.
3. Pada beban ekstrem mutlak yang digambarkan oleh baris volume 100%, permintaan terakhir yang dilayani oleh Apache2 menyentuh waktu tunggu terlama (*longest request*) yaitu sebesar 1269 ms. Di sisi lain, Nginx berhasil mengunci waktu respon terlamanya di angka 609 ms. Lonjakan ekstrem pada antrean akhir Apache2 ini merepresentasikan perlambatan performa sebesar 108.37% jauh lebih lambat dibandingkan kestabilan batas atas yang ditawarkan oleh arsitektur Nginx.

c. Dampak Struktur Arsitektur Terhadap Fluktuasi Respon Jaringan

1. Perbedaan perilaku performa yang kontradiktif antara nilai rata-rata dengan nilai puncak dipengaruhi secara langsung oleh cara masing-masing perangkat lunak server mengelola siklus hidup koneksi TCP/IP. Karakteristik pemrosesan Apache2 yang berbasis pada pembuatan *thread* mandiri (*process-per-connection*) memicu terjadinya *context switching* yang masif di tingkat *kernel* sistem operasi ketika beban koneksi simultan menyentuh angka 100 pengguna bersamaan. Hal inilah yang menyebabkan penumpukan latensi pada antrean akhir.
2. Mekanisme *non-blocking I/O* yang digerakkan oleh *event-loop* pada Nginx menjaga interupsi perangkat keras tetap minimal. Ketika antrean permintaan bertambah panjang, Nginx tidak membuang waktu komputasi untuk mengalihkan *context* antar *thread*, melainkan menyerap seluruh

koneksi ke dalam *worker process* tunggal secara asinkron. Akibatnya, distribusi latensi dari awal permintaan hingga akhir membentuk garis kurva yang stabil, linier, dan sangat mudah diprediksi

B. Implementasi Monitoring Sumber Daya Perangkat Keras

Fase evaluasi berikutnya adalah mengamati dampak operasional pemrosesan aplikasi CRUD dinamis terhadap beban kerja riil fisik perangkat keras server.(Arta et al., 2022) Perekaman data dilakukan dengan memanfaatkan utilitas pengawasan sistem real-time htop secara langsung di dalam repositori sistem operasi Ubuntu Server saat pengujian stress testing berada dalam kondisi puncaknya.

a. Analisis Beban Kerja Unit Pemroses Sentral (*CPU Utilization*)

Pengamatan terhadap persentase intensitas kerja prosesor saat paket data instruksi melintasi kartu jaringan virtual menunjukkan pola pemanfaatan daya komputasi yang berbeda antara kedua platform server. Kondisi pemanfaatan inti prosesor dapat diuraikan melalui sub-analisis berikut:

1. Selama siklus hidup eksekusi pengujian beban pada Apache2, alokasi dua inti prosesor virtual (*vCPU*) yang ditanamkan pada lingkungan mesin virtual langsung mengalami lonjakan utilisasi hingga menyentuh angka kapasitas jenuh 100% secara konstan di sepanjang waktu pengujian. Kondisi ini dipicu oleh aktivitas server yang dipaksa melakukan pembuatan (*spawning*) proses anak (*child process*) secara berulang demi mengimbangi kedatangan paket koneksi dari mesin *Host*.
2. Pada skenario pengujian Nginx yang berpasangan dengan layanan *FastCGI Process Manager* (PHP-FPM), grafik beban kerja utilitas CPU menunjukkan fluktuasi yang jauh lebih dinamis dan rileks di kisaran angka 88% hingga 96%. Prosesor tidak mengalami kejenuhan mutlak karena satu *worker process* milik Nginx mampu mengonsolidasikan antrean HTTP ke dalam satu jalur pipa pemrosesan tunggal menggunakan abstraksi *system call* epoll pada *kernel* Linux, sehingga meringankan tugas penjadwal instruksi sistem komputasi.

b. Analisis Efisiensi Manajemen Memori Utama (*RAM*)

Manajemen memori merupakan parameter penentu vital dalam mengukur efisiensi arsitektur *web server* yang dipaksa beroperasi pada infrastruktur dengan alokasi sumber daya terbatas.(Budiman et al., 2021) Berdasarkan rekaman *snapshot* utilitas sistem dari gambar htop yang diambil, kondisi penggunaan kapasitas RAM dipetakan secara komparatif pada Tabel 4.

Tabel 4. Perbandingan Konsumsi Memori Utama Server

Status Kondisi Operasional Sistem	Konsumsi RAM Nginx + PHP-FPM	Konsumsi RAM Apache2
Idle Status (Tanpa Beban)	645 MB	645 MB
Peak Status (Beban Puncak)	712 MB	894 MB
Net Volatility (Lonjakan Bersih)	67 MB	249 MB

Penjelasan komprehensif dari komparasi manajemen alokasi memori pada Tabel 4 dijabarkan melalui poin-poin analisis berikut:

1. Status *Idle* menunjukkan *baseline* penggunaan memori sistem operasi Ubuntu Server 22.04 LTS yang bersih sebelum menerima paket trafik, di mana kedua server berada pada titik awal yang sama yaitu sebesar 645 MB. Angka ini mencakup kebutuhan dasar proses sistem dasar dan layanan basis data MySQL 8.0 yang berjalan aktif di latar belakang.

2. Pada saat instruksi pengujian dijalankan dari mesin *Host* hingga mencapai *Peak Status*, konsumsi memori pada server Apache2 membengkak secara instan hingga menyentuh angka 894 MB. Kondisi ini menghasilkan nilai lonjakan bersih (*Net Volatility*) memori yang terserap sebesar 249 MB. Hal ini membuktikan bahwa arsitektur berbasis proses memerlukan alokasi memori yang tinggi akibat setiap *thread* baru wajib membawa salinan konfigurasi global server secara mandiri.
3. Pada pengujian tandingan, Nginx bersama PHP-FPM hanya menaikkan total konsumsi memori server ke angka 712 MB saat beban puncak berlangsung. Lonjakan bersih memori yang dihasilkan sangat minim, yaitu hanya sebesar 67 MB. Jika dibandingkan secara langsung, manajemen memori Nginx jauh lebih efisien sebesar 73.09% dalam menghemat ruang penyimpanan RAM dibandingkan dengan manajemen memori yang dimiliki oleh Apache2.

C. Pembahasan Teoretis dan Implikasi Praktis Infrastruktur

Integrasi seluruh data eksperimen kuantitatif menghasilkan sebuah pemahaman teoritis baru mengenai karakteristik operasional kedua *web server* raksasa ini ketika dipaksa bekerja di dalam ruang virtual yang sempit dan terbatas.(Amrullah et al., 2023)

a. Pemecahan Paradoks Kecepatan Rata-rata dan Perlambatan Beban Puncak

Temuan yang menunjukkan bahwa Apache2 memiliki RPS yang lebih tinggi namun memiliki latensi puncak yang buruk memunculkan sebuah pemahaman tekniths yang mendalam:

1. Keunggulan RPS Apache2 (225.13 RPS) didorong oleh kemampuan manajemen *prefork/event* bawaan Ubuntu yang sangat agresif dalam memanfaatkan seluruh ketersediaan daya prosesor sejak milidetik pertama pengujian dimulai. Interaksi langsung antara server dan database terjadi secara instan tanpa perantara protokol komunikasi tambahan.
2. Namun, agresivitas ini mengorbankan stabilitas ketika antrean masuk ke dalam kondisi antrean panjang (*long queue*). Ketika 100 koneksi bersamaan menghantam sistem, alokasi memori yang membengkak hingga 894 MB (seperti tertera pada Tabel 4) memicu terjadinya fenomena *thrashing* ringan pada sistem operasi. Akibatnya, beberapa paket permintaan terakhir harus mengantre dalam durasi yang jauh lebih lama, memicu lonjakan latensi ekstrem hingga 1269 ms pada Tabel 3. Nginx menghindari paradoks ini dengan mengorbankan sedikit kecepatan rata-rata demi menjamin keadilan distribusi waktu tunggu bagi setiap paket data yang masuk ke sistem.

Meskipun pengujian komparatif ini tidak mengimplementasikan uji statistik inferensial formal (seperti *Independent T-Test*), validitas dan signifikansi perbedaan performa tetap terjaga secara konsisten melalui replikasi *stress testing* yang menghasilkan nilai *Failed Requests* mutlak nol.(Wicoksono et al., 2025) Namun, perlu dicatat bahwa eksperimen ini memiliki keterbatasan ruang lingkup karena dijalankan di dalam lingkungan virtualisasi VirtualBox dengan alokasi sumber daya yang terisolasi. Pada infrastruktur *bare-metal* atau *cloud server* produksi nyata, fenomena *thrashing* dan *context switching* pada Apache2 mungkin dapat diredam lebih baik oleh algoritma penjadwalan kernel hulu, sehingga hasil komparasi ini secara spesifik merepresentasikan performa pada arsitektur komputer virtual bersumber daya minimalis

b. Rekomendasi Desain Infrastruktur Hasil Eksperimen

Berdasarkan seluruh hasil analisis performa dan efisiensi konsumsi sumber daya perangkat keras yang telah dibedah, rancangan implementasi praktis bagi

pengembang sistem dapat dirumuskan melalui poin-poin keputusan berikut:

1. Penggunaan arsitektur Apache2 sangat direkomendasikan pada skenario sistem aplikasi web internal kampus atau perusahaan berskala menengah, di mana jumlah pengguna simultan dapat diprediksi dengan pasti dan karakteristik aplikasi menuntut kecepatan respon individual yang tinggi untuk eksekusi fungsi-fungsi kalkulasi data yang kompleks.
2. Penggunaan arsitektur Nginx yang dipadukan dengan PHP-FPM mutlak digunakan pada skenario sistem aplikasi publik yang menghadapi pola trafik fluktuatif dan berpotensi menerima lonjakan pengunjung tak terduga secara masif, seperti portal penerimaan mahasiswa baru atau sistem ujian daring. Kemampuan Nginx dalam mengunci latensi maksimal di bawah 609 ms menjamin aplikasi tetap responsif dan terhindar dari resiko kegagalan sistem (*service unavailable*) akibat kehabisan alokasi memori utama RAM.

Temuan dalam penelitian ini memperkuat kajian terdahulu oleh (Kavitha & Varalakshmi, 2018)dan (Yang et al., 2020) yang menyatakan bahwa arsitektur *event-driven* milik Nginx selalu menawarkan skalabilitas manajemen memori yang superior pada kondisi trafik tinggi dibandingkan model *process-based*. Namun, keunikan atau kebaruan hasil dari eksperimen ini menunjukkan bahwa pada skenario CRUD dinamis berskala menengah di lingkungan virtual, Apache2 masih mampu memberikan laju pemrosesan transaksi (*throughput*) yang sedikit lebih agresif di awal sesi pengujian sebelum mencapai titik jenuh pasokan RAM, sebuah karakteristik yang melengkapi pemetaan performa pada studi-studi infrastruktur server sebelumnya.

SIMPULAN (PENUTUP)

Berdasarkan seluruh rangkaian eksperimen laboratorium terkontrol dan analisis data komparatif yang telah dilakukan, dapat disimpulkan bahwa pemilihan arsitektur web server memiliki dampak yang sangat krusial terhadap performa aplikasi CRUD dinamis berbasis PHP dan MySQL di lingkungan virtual Ubuntu Server. Penelitian ini berhasil menjawab permasalahan utama terkait degradasi kinerja server pada kondisi trafik padat, di mana kedua web server menunjukkan karakteristik operasional yang bertolak belakang untuk menangani interaksi basis data dinamis.

Dari aspek kuantitas pemrosesan data, Apache2 dengan arsitektur tradisional berbasis proses terbukti unggul dengan menghasilkan throughput sebesar 225,13 Requests per Second (RPS). Angka performa tersebut menjadikan Apache2 sekitar 8,48% lebih cepat dibandingkan Nginx dalam menyelesaikan siklus permintaan rata-rata pada kondisi beban kerja konstan di skala menengah.

Namun, ketika sistem diuji pada kondisi beban puncak ekstrem yang melibatkan 100 koneksi simultan, Nginx dengan arsitektur event-driven mampu menyelesaikan masalah stabilitas layanan secara signifikan. Nginx berhasil mengunci batas atas latensi maksimal hanya pada angka 609 ms, sementara Apache2 mengalami lonjakan latensi hingga 1269 ms yang memicu perlambatan respon bagi pengguna akhir.

Efisiensi Nginx ini diperkuat oleh pengamatan utilitas sistem di mana manajemen memorinya jauh lebih hemat sebesar 73,09% dalam menyerap alokasi RAM dibandingkan Apache2 saat beban puncak berlangsung. Dengan demikian, Nginx merupakan solusi paling optimal untuk menjaga konsistensi waktu respon dan skalabilitas aplikasi pada infrastruktur virtual dengan sumber daya yang terbatas.

Akan tetapi, perlu ditegaskan kembali bahwa seluruh kesimpulan performa dalam penelitian ini secara spesifik berlaku pada skenario pengujian aplikasi CRUD berbasis PHP-MySQL di dalam ekosistem virtualisasi VirtualBox dengan batasan alokasi sumber daya

tertentu, sehingga karakteristik data yang diperoleh tidak dapat langsung digeneralisasikan secara mutlak pada seluruh arsitektur *web server* di lingkungan produksi nyata.

UCAPAN TERIMA KASIH

Penulis menyampaikan terima kasih yang sebesar-besarnya kepada program studi Teknologi Informasi beserta seluruh jajaran dosen yang telah memberikan dukungan akademik serta fasilitas laboratorium komputer selama proses penyusunan penelitian ini. Terima kasih juga penulis sampaikan kepada rekan-rekan sejawat dan semua pihak yang telah memberikan saran serta kritik membangun demi kesempurnaan artikel ilmiah ini.

DAFTAR PUSTAKA

Amrullah, A., Nugroho, A., & Ramadhan, Z. (2023). Perbandingan Kinerja Web Server Pada Penyedia Layanan Cloud Microsoft Azure Dan Amazon Web Services. *Jurnal Ilmiah Komputasi*, *5*(1), 92–97. https://pdfs.semanticscholar.org/056c/57b669a3ce3d92e8b41f6fc0a3c4c913293f.pdf

Arta, Y., Wandri, R., Hanafiah, A., Kristian Pranoto, B., & Rizki Fadhilah, M. (2022). Analisa Perbandingan Web Server Untuk Kebutuhan Open Journal System (OJS) Menggunakan Secure Tunnel. *Cogito Smart Journal*, *8*(2). https://cogito.unklab.ac.id/index.php/cogito/article/view/407

Azi, M. N. A., Arifwidodo, B., & Wahyudi, E. (2023). Analisis Performansi Web Server Saat Menangani Permintaan Client Menggunakan Metode Reserve Proxy Caching dan Varnish. *Jurnal Telekomunikasi Dan Elektronika (JTECE)*, *5*(1), 92–97. http://journal.ittelkom-pwt.ac.id/index.php/jtece/article/view/843

Budiman, S., Saputro, V. A., & Fadhila, F. (2021). Analisis Performa Kecepatan MySQL dan NoSQL Pada Sistem Operasi Windows dan Linux. *Jurnal Jnanaloka*. https://lenteradua.net/jurnal/index.php/jnanaloka/article/view/29

Faisal, & Aziz, A. S. (2025). Analisis Kinerja Web Server Apache, Nginx, Open Litespeed, Dan Open Resty. *Journal of Information Computer Science (JICS)*, *11*(1). https://jurnal.uui.ac.id/index.php/jics/article/view/4744

Handayani, D., Rizky Maulana Ibrahim, S., Eka Valentina, P., & Maulana Kusumah Putra, F. (2026). Komparasi Performa REST API Laravel 11 dan CodeIgniter 4 Menggunakan Metode Eksperimental. *Jurnal Ilmu Komputer (JIK)*. https://ejurnal.umri.ac.id/index.php/JIK/article/view/11274

Kavitha, B., & Varalakshmi, P. (2018). Performance analysis of virtual machines and docker containers. *Lecture Notes in Networks and Systems*, *804*, 99–113. https://doi.org/10.1007/978-981-10-8603-8_9

Lohja, G., Kor, A., & Jawad, N. (2023). *Comparative Analysis of Energy Consumption and Carbon Footprint of Apache, Nginx, and Lighttpd Web Servers*. https://figshare.leedsbeckett.ac.uk/articles/conference_contribution/Comparative_Analysis_of_Energy_Consumption_and_Carbon_Footprint_of_Apache_Nginx_and_Lighttpd_Web_Servers/29100068/1

Luthfi, M., Data, M., & Yahya, W. (2018). Perbandingan performa reverse proxy caching nginx dan varnish pada web server apache. *Jurnal Pengembangan Teknologi Informasi Dan Ilmu Komputer (J-PTIIK)*, *2*(4), 1457–1463. https://j-ptiik.ub.ac.id/index.php/j-ptiik/article/view/1206

Maila, H., Indra, D., & Satra, R. (2020). Analisis Perbandingan Layanan Data Server Menggunakan Failover Cluster pada Platform Nginx dan Apache.

Jurnal Buletin Sistem Informasi Dan Teknologi (BUSITI). https://jurnal.fikom.umi.ac.id/index.php/BUSITI/article/view/829

Nurazizah, R. A. S., Yustiana, I., & Hermanto. (2024). Evaluasi Website Open Data Kota Sukabumi Berdasarkan Standar ISO/IEC 9126. *Jurnal Ilmiah Pendidikan Informatika (JIPI)*, *9*(1), 79–89. https://doi.org/10.29100/jipi.v9i1.4281

Rafli, M., Fitri, I., & Andrianingsih, A. (2022). Jurnal Pengujian Kinerja Load Balancing Web Server menggunakan Nginx Riverse Proxy Berbasis OS Centos 7. *Jurnal Teknologi Informasi Dan Sistem Informasi (JATISI)*, *9*(3), 1824–1840. https://jurnal.mdp.ac.id/index.php/jatisi/article/view/2185

Satwika, I. K. S., & Semadi Ketut Ngurah. (2020). Evaluasi Kinerja Web Server Apache menggunakan Protokol HTTP2. *Jurnal Ekonomi, Teknologi, Dan Akuntansi (JETAS)*. https://scholar.archive.org/work/xwateyefbfgb5aq54hga3loori/access/wayback/http://lamintang.org/journal/index.php/jetas/article/download/92/68/

Warman, I., & Nugraha, M. Y. S. (2017). Performa Kinerja Web Server Berbasis Ubuntu Linux Dan Turnkey Linux. *Jurnal Teknik Informatika (JTIF)*, *5*(2). https://doi.org/10.21063/JTIF.2017.V5.2.63-72

Wicoksono, W., Mustaqhim, H. A., Anwas, P. P., & Badratul, L. N. L. (2025). Performance comparison of NGINX, Apache, and Lighttpd using WRK on a Debian. *Bit-Tech Journal*. https://jurnal.kdi.or.id/index.php/bt/article/view/2661

Yang, S., Wang, X., Wang, X., An, L., & Zhang, G. (2020). Performance evaluation of docker container and virtual machine. *World Wide Web*, *23*(4), 2593–2632. https://doi.org/10.1007/S11280-020-00789-9

Zulfahrizan, A., Raffi Akbar Tanjung, M., Zidane Al-Kautsar, M., Kiswanto, D., Willem Iskandar, J., Medan Estate, P. V, Sei Tuan, P., & Serdang, D. (2025). Comparative of Performance Optimization Techniques for PHP Framework Testing. *Brilliance: Research of Artificial Intelligence*, *9*(1). https://itscience-indexing.com/jurnal/index.php/brilliance/article/view/5989