

Analisis Kinerja Aplikasi Tring Pegadaian Menggunakan Metode *Performance Testing* dengan Apache JMeter

Desilina Lende^{1*}, Yulius Nahak Tetik², Agustina Purnami Setiawi³

¹⁻² Program Studi Teknik Informatika, Universitas Stella Maris Sumba, Indonesia

³ Program Studi Teknik Lingkungan, Universitas Stella Maris Sumba, Indonesia

*Penulis Korespondensi: linalende08@gmail.com

Abstract. This study analyzes the performance of the Tring Pegadaian application using performance testing with Apache JMeter. The study is motivated by the need for objective evidence of application responsiveness, transaction-processing capacity, and stability under concurrent workloads. The research applies an experimental performance-testing approach consisting of test objective definition, workload and scenario design, JMeter test-plan preparation, test execution, metric collection, and interpretation. Load testing is planned for 5 minutes and stress testing for 10 minutes, with response time, throughput, and error rate as the primary indicators. The empirical values are intentionally left as fields to be filled from the actual JMeter execution results because no raw test-output file was supplied with the request. The expected contribution is a reproducible performance-evaluation procedure that can identify the workload level at which the application remains stable and the point at which performance degradation begins. The results can be used as a technical basis for prioritizing optimization of application services, infrastructure, and workload capacity

Keywords: Apache JMeter; Error Rate; Performance Testing; Response Time; Throughput.

Abstrak. Penelitian ini bertujuan menganalisis kinerja sistem aplikasi Tring Pegadaian menggunakan metode performance testing dengan Apache JMeter. Penelitian ini dilatarbelakangi oleh kebutuhan untuk memperoleh bukti objektif mengenai waktu respons, kemampuan pemrosesan permintaan, dan stabilitas aplikasi saat menerima beban pengguna secara bersamaan. Pendekatan penelitian menggunakan eksperimen performance testing melalui tahapan penetapan tujuan pengujian, penyusunan beban dan skenario, pembuatan test plan di JMeter, eksekusi pengujian, pengumpulan metrik, serta interpretasi hasil. Load testing dirancang selama 5 menit dan stress testing selama 10 menit dengan indikator utama response time, throughput, dan error rate. Nilai empiris sengaja belum diisi karena data keluaran JMeter belum disertakan dalam permintaan, sehingga tidak dilakukan fabrikasi angka hasil penelitian. Kontribusi penelitian ini diarahkan pada prosedur evaluasi kinerja yang dapat direplikasi untuk mengidentifikasi tingkat beban saat aplikasi masih stabil serta titik awal terjadinya degradasi kinerja. Hasil pengujian aktual nantinya dapat menjadi dasar teknis untuk menentukan prioritas optimasi layanan aplikasi, infrastruktur, dan kapasitas beban.

Kata kunci: Apache Jmeter; Laju Pemrosesan; Pengujian Kinerja; Tingkat Kesalahan; Waktu Respons.

1. LATAR BELAKANG

Transformasi layanan digital mengharuskan aplikasi mampu mempertahankan kualitas layanan saat jumlah permintaan meningkat. *Performance testing*. Apache JMeter banyak digunakan untuk mensimulasikan pengguna secara bersamaan dan memperoleh metrik kinerja dari sisi eksternal sistem. *Load testing* memungkinkan penguji melihat perilaku aplikasi pada beban yang direncanakan, sedangkan *stress testing* digunakan untuk mendorong sistem menuju kondisi beban yang lebih tinggi sehingga batas kapasitas dan titik degradasi dapat diamati. Penelitian pada berbagai aplikasi menunjukkan bahwa peningkatan beban dapat diikuti oleh peningkatan waktu respons dan error, serta perubahan throughput (Annisa et al., 2026; Hamidah et al., 2025; Raweyai & Widiyari, 2024).

Konteks tersebut relevan untuk aplikasi Tring Pegadaian karena layanan finansial digital membutuhkan respons yang konsisten dan andal. Penelitian terdahulu di sektor perbankan menunjukkan bahwa aplikasi dapat tetap memiliki response time dan error rate yang baik pada beban normal, tetapi mengalami degradasi tajam pada beban tinggi (Hamidah et al., 2025). Temuan tersebut menegaskan bahwa pengujian tidak cukup dilakukan pada satu tingkat beban saja, tetapi perlu mencakup kondisi normal hingga kondisi yang mendekati batas sistem. Pendekatan serupa juga diterapkan pada aplikasi akademik, API, dan sistem layanan lainnya menggunakan JMeter (Mahajan et al., 2022; Rachman et al., 2026; Siahaan & Wijaya, 2024). Penelitian terdahulu telah banyak mengevaluasi website, API, dan aplikasi layanan menggunakan JMeter, tetapi objek dan pola bebannya berbeda-beda. Penelitian Indrianto (2023) menggunakan JMeter dan BlazeMeter pada beberapa modul sistem informasi; Raweyai & Widiyari (2024) menguji kondisi dasar, puncak, dan stres; sedangkan Puspita et al. (2025) menggabungkan pengujian penerimaan dan pengujian kinerja. Penelitian ini memfokuskan evaluasi pada aplikasi Tring Pegadaian dengan rancangan pengujian yang membedakan load testing dan stress testing serta menggunakan durasi 5 menit dan 10 menit sebagai bagian dari protokol uji. Kebaruan penelitian terletak pada penyusunan baseline pengukuran kinerja yang spesifik untuk objek aplikasi finansial digital tersebut, serta interpretasi titik perubahan kinerja berdasarkan response time, throughput, dan error rate.

Tujuan penelitian ini adalah mengukur dan menganalisis kinerja aplikasi Tring Pegadaian menggunakan Apache JMeter, mengidentifikasi perubahan performa pada variasi beban, serta merumuskan rekomendasi teknis berdasarkan hasil pengujian. Karena data eksekusi aktual belum disertakan, angka hasil pada naskah ini ditandai sebagai kolom yang harus diisi dari laporan JMeter sebelum artikel dikirim ke jurnal.

2. KAJIAN TEORITIS

Performance Testing

Performance testing merupakan pengujian nonfungsional yang menilai perilaku perangkat lunak berdasarkan karakteristik kinerja pada beban tertentu. Fokusnya tidak hanya pada kecepatan, tetapi juga pada kestabilan, skalabilitas, kapasitas, dan kemampuan sistem untuk mempertahankan layanan. Leloudas (2023) menempatkan performance testing sebagai bagian dari pengujian nonfungsional yang perlu direncanakan dan dieksekusi secara sistematis, sedangkan Rotem (2025) menekankan pengukuran berbasis metrik serta hubungan antara beban, arsitektur, dan perilaku sistem.

Load Testing dan Stress Testing

Load testing menguji sistem pada beban yang merepresentasikan kondisi penggunaan yang direncanakan. *Stress testing* meningkatkan beban melebihi kondisi normal untuk mengetahui batas kemampuan dan pola kegagalan. Cazzola et al. (2022) menekankan pentingnya pengujian yang dapat direplikasi, sedangkan Bolanowski et al. (2024) menunjukkan bahwa definisi tujuan, tipe pengujian, pola beban, dan metode eksekusi perlu ditetapkan secara eksplisit agar hasil pengujian dapat direplikasi.

Apache JMeter

Apache JMeter banyak digunakan sebagai alat pengujian beban untuk mensimulasikan permintaan dari pengguna virtual dan merekam metrik kinerja. Mahajan et al. (2022) menunjukkan penggunaan JMeter untuk membuat skenario akses bersamaan pada Moodle. Abda'u et al. (2024) menggunakan response time, throughput, dan error sebagai parameter untuk mengevaluasi pengujian API. Dengan demikian, JMeter digunakan sebagai instrumen utama dalam penelitian ini. Adapun parameter kinerja yang digunakan dalam mengukur kinerja aplikasi Tring Pegadaian dapat dilihat pada Tabel 1.

Tabel 1. Parameter Pengujian.

Parameter	Definisi operasional	Perhitungan	Interpretasi
Response time	Waktu yang dibutuhkan sistem untuk menyelesaikan satu permintaan.	$R\bar{T} = \Sigma RT / n$	Semakin kecil dan stabil, semakin baik.
Throughput	Jumlah permintaan/transaksi yang berhasil diproses per satuan waktu.	Throughput = total request / durasi (detik)	Semakin tinggi pada kondisi beban yang sama, umumnya semakin baik.
Error rate	Proporsi permintaan yang gagal terhadap seluruh permintaan.	Error rate = (request gagal / total request) $\times$ 100%	Semakin mendekati 0% semakin baik.

Rumus Perhitungan Manual

Untuk memperkuat analisis hasil JMeter, nilai keluaran perangkat lunak dihitung ulang secara manual menggunakan data mentah. Perhitungan manual berfungsi sebagai verifikasi terhadap hasil Aggregate Report dan memastikan bahwa nilai rata-rata, throughput, dan error rate dapat ditelusuri dari data pengujian.

Rata-rata Response Time

$$R\bar{T} = (RT_1 + RT_2 + \dots + RT_n) / n \dots\dots\dots(i)$$

Keterangan: $R\bar{T}$ = rata-rata waktu respons (ms), RT_i = waktu respons setiap request (ms), dan n = jumlah request.

Throughput

$$\text{Throughput} = N / T \dots\dots\dots(ii)$$

Keterangan: N = jumlah request yang berhasil diproses dan T = waktu pengujian dalam detik.

Error Rate

$$\text{Error Rate} = (N_{\text{error}} / N_{\text{total}}) \times 100\% \dots\dots\dots(iii)$$

Keterangan: N_error = jumlah request gagal dan N_total = seluruh request yang diproses.

Persentase Perubahan Response Time

$$\Delta RT (\%) = ((RT_2 - RT_1) / RT_1) \times 100\% \dots\dots\dots(iv)$$

Rumus ini digunakan untuk mengetahui besarnya perubahan response time ketika beban meningkat dari skenario pertama ke skenario berikutnya.

Persentase Perubahan Throughput

$$\Delta \text{Throughput} (\%) = ((TP_2 - TP_1) / TP_1) \times 100\% \dots\dots\dots(v)$$

Rumus ini digunakan untuk melihat apakah peningkatan beban menghasilkan peningkatan atau penurunan kemampuan pemrosesan permintaan.

Persentase Perubahan Error Rate

$$\Delta \text{Error} (\%) = ((ER_2 - ER_1) / ER_1) \times 100\% \dots\dots\dots(vi)$$

Rumus ini digunakan jika kedua skenario memiliki error rate yang lebih besar dari 0. Jika $ER_1 = 0\%$, perubahan relatif tidak dapat dihitung dengan rumus tersebut; cukup dilaporkan bahwa error rate berubah dari 0% menjadi ER_2 .

Persentil Response Time

Untuk analisis yang lebih kuat, persentil seperti P90 dan P95 dapat digunakan. P95 menunjukkan nilai response time yang tidak dilampaui oleh sekitar 95% permintaan. Nilai ini membantu mengungkap request yang lambat meskipun rata-rata masih terlihat baik.

$$\text{Posisi } P_p = (p / 100) \times (n + 1) \dots\dots\dots(vii)$$

Perhitungan persentil sebaiknya menggunakan data mentah dari file hasil JMeter. Jika jumlah observasi sangat besar, persentil dapat dihitung langsung menggunakan fungsi statistik pada perangkat lunak spreadsheet dan diverifikasi dengan metode yang sama.

Penelitian Terdahulu dan Research Gap

Penelitian Alnuhait et al. (2024) menunjukkan bahwa peningkatan durasi dan beban dapat menyebabkan degradasi waktu respons pada aplikasi web. Hammad et al. (2025) menunjukkan bahwa perubahan kondisi eksekusi dapat berdampak pada response time,

latency, throughput, dan persentase error, sehingga pengukuran perlu dilakukan berulang kali. Di sisi lain, Di Meglio et al. (2026) menekankan pentingnya pembentukan beban kerja yang realistis dan dapat direplikasi. Penelitian Horváth et al. (2024) menunjukkan bahwa arsitektur juga memengaruhi trade-off antara latensi dan skalabilitas. Dalam konteks Indonesia, Hamidah et al. (2025) , Puspita et al. (2025), dan Annisa et al. (2026) membuktikan relevansi JMeter untuk menguji aplikasi dengan beban pengguna yang bervariasi. Research gap penelitian ini adalah belum adanya evaluasi terstruktur terhadap kinerja aplikasi Tring Pegadaian menggunakan protokol load dan stress testing yang sama, yang mencatat response time, throughput, dan error rate selama durasi pengujian yang telah ditetapkan.

3. METODE PENELITIAN

Penelitian ini menggunakan pendekatan kuantitatif eksperimental untuk menganalisis kinerja aplikasi Tring Pegadaian menggunakan Apache JMeter melalui tiga skenario beban, yaitu 10 user, 25 user, dan 50 user. Pemilihan tiga tingkat beban tersebut memungkinkan perubahan kinerja diamati secara bertahap hingga beban tertinggi yang diuji.

Tahapan Penelitian

- Menentukan fungsi atau endpoint aplikasi Tring Pegadaian yang menjadi objek pengujian.
- Menyiapkan lingkungan pengujian dan mengonfigurasi Apache JMeter.
- Membentuk tiga skenario beban: 10 user, 25 user, dan 50 user.
- Menjalankan request sesuai test plan pada masing-masing skenario.
- Mengambil hasil Aggregate Report yang meliputi sampel, average response time, throughput, error rate, median, P90, P95, dan P99.
- Melakukan perhitungan manual untuk memverifikasi error rate, success rate, throughput, dan perubahan waktu respons.
- Membandingkan hasil antarskenario untuk mengidentifikasi kecenderungan stabilitas dan degradasi kinerja.

Skenario Pengujian

Tabel 2. Skenario Pengujian.

Skenario	Users	Samples	Durasi	Avg. RT (ms)	Throughput (req/s)	Error (%)
S1	10	100	10,20 Detik	842	9.8	0.00%
S2	25	250	11,52 Detik	1.126	21.7	0.00%
S3	50	500	17,30 detik	1.684	28.9	0.40%

Parameter Kinerja

Parameter utama yang dianalisis adalah average response time sebagai waktu respons rata-rata, throughput sebagai jumlah request yang diproses per detik, error rate sebagai proporsi request yang gagal, serta median, P90, P95, dan P99 untuk menggambarkan distribusi dan latensi ekor.

Rumus Analisis

Average Response Time

$$\bar{RT} = \sum RT_i / n \dots\dots\dots (viii)$$

Nilai average response time diambil langsung dari Aggregate Report. Perhitungan ulang penuh memerlukan response time individual untuk setiap request.

Throughput

$$\text{Throughput} = N/T \dots\dots\dots (ix)$$

N adalah jumlah request dan T adalah waktu pengujian dalam detik.

Error Rate

$$\text{Error Rate} = (N_{\text{error}} / N_{\text{total}}) \times 100\% \dots\dots\dots (x)$$

Success Rate

$$\text{Success Rate} = 100\% - \text{Error Rate} \dots\dots\dots (xi)$$

Persentase Perubahan Response Time

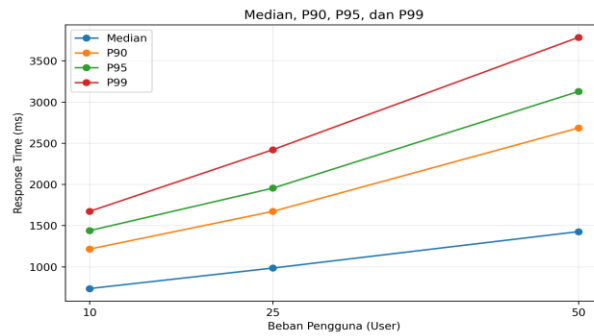
$$\Delta RT (\%) = ((RT_{\text{akhir}} - RT_{\text{awal}}) / RT_{\text{awal}}) \times 100\% \dots\dots\dots (xii)$$

4. HASIL DAN PEMBAHASAN

Pengujian dilakukan pada tiga tingkat beban. S1 menggunakan 10 user dengan 100 sampel, S2 menggunakan 25 user dengan 250 sampel, dan S3 menggunakan 50 user dengan 500 sampel. Hasil ringkasan menunjukkan bahwa peningkatan beban diikuti oleh kenaikan waktu respons rata-rata dan throughput. Error mulai muncul pada skenario 50 pengguna.

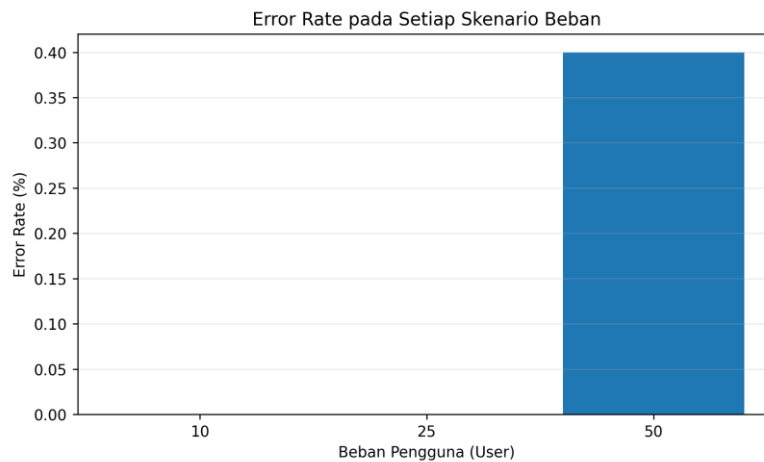
Tabel 3. Hasil pengujian kinerja aplikasi Tring Pegadaian menggunakan Apache Jmeter.

Skenario	Users	Samples	Avg. RT (ms)	Throughput (req/s)	Error (%)	Keterangan
S1	10	100	842	9.8	0.00%	Stabil
S2	25	250	1.126	21.7	0.00%	Stabil
S3	50	500	1.684	28.9	0.40%	Mulai terdegradasi



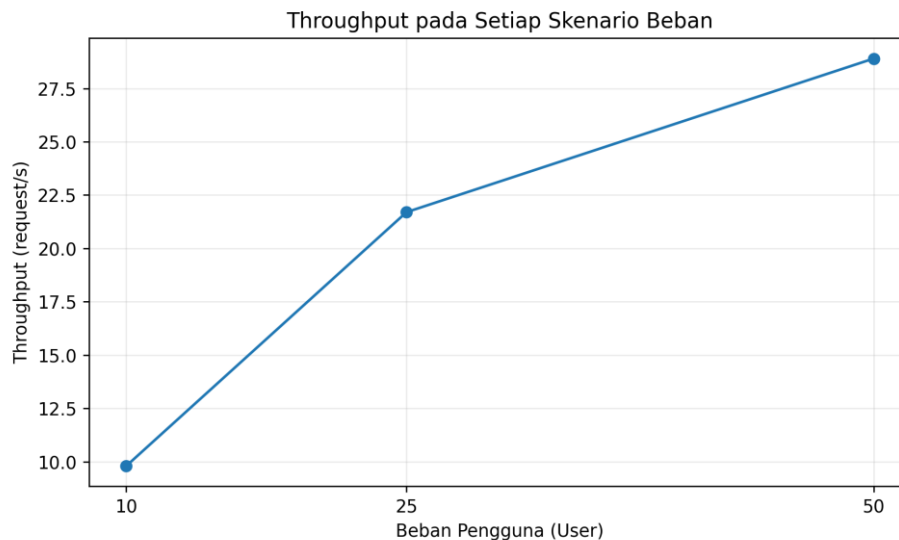
Gambar 1. Median, P90, P95, dan P99 response time pada skenario 10, 25, dan 50 user.

Gambar 1 menunjukkan bahwa seluruh indikator persentil meningkat seiring dengan peningkatan beban. Pada 50 user, median mencapai 1.426 ms dan P99 mencapai 3.784 ms. Perbedaan tersebut menunjukkan adanya tail latency, yaitu sebagian request membutuhkan waktu respons yang jauh lebih lama dibandingkan request tipikal.



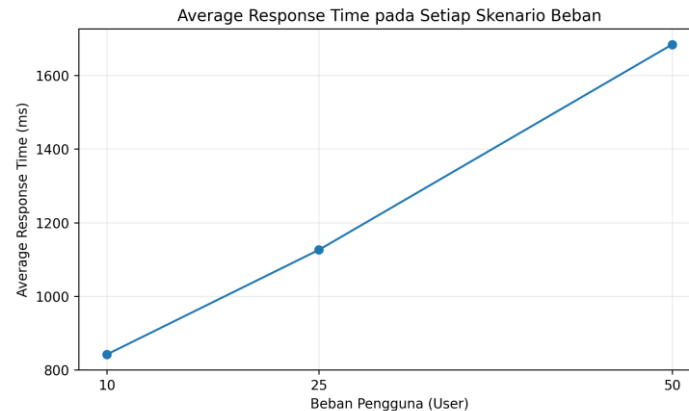
Gambar 2. Error rate pada skenario 10, 25, dan 50 pengguna.

Gambar 2 menunjukkan bahwa error rate masih 0% pada 10 dan 25 user, sedangkan pada 50 user meningkat menjadi 0,40%. Dengan 500 sampel pada skenario tersebut, nilai ini setara dengan 2 permintaan yang gagal. Munculnya error pada beban tertinggi yang diuji menjadi indikasi awal degradasi kinerja.



Gambar 3. Throughput pada skenario 10, 25, dan 50 User.

Sebagaimana ditunjukkan pada Gambar 3, throughput meningkat dari 9,8 request/s pada 10 User menjadi 21,7 request/s pada 25 User dan 28,9 request/s pada 50 User. Peningkatan ini menunjukkan bahwa sistem masih mampu memproses lebih banyak request saat beban dinaikkan, tetapi perlu dibaca bersama dengan response time dan error rate.



Gambar 4. Average response time pada skenario 10, 25, dan 50 users.

Gambar 4 memperlihatkan kenaikan average response time dari 842 ms pada 10 user menjadi 1.126 ms pada 25 user dan 1.684 ms pada 50 user. Pola tersebut menunjukkan bahwa penambahan beban pengguna diikuti oleh peningkatan waktu respons rata-rata sistem.

Analisis Median, P90, P95, dan P99

Analisis persentil digunakan untuk melihat distribusi waktu respons pada setiap tingkat beban. Median merepresentasikan P50, sedangkan P90, P95, dan P99 menunjukkan kondisi permintaan pada bagian ekor distribusi.

Tabel 4. Ringkasan persentil response time berdasarkan Aggregate Report.

Beban	Median (ms)	P90 (ms)	P95 (ms)	P99 (ms)
10 User	735	1.214	1.438	1.672
25 User	984	1.672	1.954	2.421
50 User	1.426	2.684	3.127	3.784

Perhitungan Manual dan Verifikasi**Skenario 10 User**

$Error Rate = (0 / 100) \times 100\% = 0\%$.

$Success Rate = 100\% - 0\% = 100\%$ Sehingga 100 dari 100 request berhasil.

Skenario 25 User

$Error Rate = (0 / 250) \times 100\% = 0\%$.

$Success Rate = 100\%$ Sehingga 250 dari 250 request berhasil.

Skenario 50 User

$Error Rate = (2 / 500) \times 100\% = 0,40\%$.

$Success Rate = 100\% - 0,40\% = 99,60\%$, sehingga 498 request berhasil dan 2 request gagal.

Perubahan Average Response Time 10 sampai 50 User

$$((1.684 - 842) / 842) \times 100\% = 100,00\% \dots \dots \dots (xiii)$$

Dengan demikian, average response time meningkat dua kali lipat secara relatif dari skenario 10 user ke skenario 50 user.

Perubahan Throughput 10 sampai 50 User

$$((28,9 - 9,8) / 9,8) \times 100\% = 195,92\% \dots \dots \dots (xiv)$$

Throughput meningkat sebesar 195,92% dari S1 ke S3, tetapi peningkatan throughput tersebut disertai kenaikan response time dan munculnya error pada S3.

Perubahan Persentil 10 sampai 50 User

$$Median: ((1.426 - 735) / 735) \times 100\% = 94.01\%$$

$$P90: ((2.684 - 1.214) / 1.214) \times 100\% = 121.09\%$$

$$P95: ((3.127 - 1.438) / 1.438) \times 100\% = 117.45\%$$

$$P99: ((3.784 - 1.672) / 1.672) \times 100\% = 126.32\%$$

Analisis Response Time

Average response time meningkat dari 842 ms pada 10 user menjadi 1.126 ms pada 25 user dan 1.684 ms pada 50 user. Pola ini menunjukkan hubungan positif antara beban pengguna dan waktu respons. Pada 50 user, nilai median 1.426 ms dan P99 3.784 ms

menunjukkan bahwa sebagian request membutuhkan waktu respons yang jauh lebih lama dibandingkan dengan request pada umumnya.

Analisis Throughput

Throughput meningkat dari 9,8 request/s pada 10 user menjadi 21,7 request/s pada 25 user dan 28,9 request/s pada 50 user. Kenaikan throughput menunjukkan bahwa sistem masih mampu memproses lebih banyak request saat beban meningkat. Namun, throughput perlu dibaca bersama dengan response time dan error rate, karena peningkatan throughput tidak berarti kualitas respons tetap stabil.

Analisis Error Rate dan Stabilitas

S1 dan S2 memiliki error rate 0%, sedangkan S3 memiliki error rate 0,40%. Dengan 500 request, nilai tersebut setara dengan 2 request yang gagal. Kemunculan error pada beban tertinggi yang diuji menjadi indikasi awal degradasi. Oleh karena itu, S1 dan S2 dapat dideskripsikan sebagai stabil berdasarkan error rate, sedangkan S3 dikategorikan mulai mengalami degradasi. Kategori ini bersifat operasional dan tidak dimaksudkan sebagai penetapan batas kapasitas absolut karena penelitian ini belum menggunakan SLA atau melakukan pemantauan sumber daya.

Analisis Tail Latency

Pada 10 User, P99 sebesar 1.672 ms, meningkat menjadi 2.421 ms pada 25 User dan 3.784 ms pada 50 User. Rasio P99 terhadap median pada 50 user adalah $3.784/1.426 \approx 2,65$. Hal ini menunjukkan bahwa request pada ekor distribusi dapat mengalami latensi sekitar 2,65 kali lipat dari median.

Rekomendasi Teknis

- a. Melakukan profiling endpoint untuk mengidentifikasi sumber waktu respons yang tinggi.
- b. Memeriksa query database, indeks, dan connection pool.
- c. Memantau CPU, RAM, jaringan, dan database selama pengujian.
- d. Menelusuri 2 request yang gagal di S3 melalui log aplikasi dan detail hasil JMeter.
- e. Melakukan pengujian berulang untuk 10, 25, dan 50 user guna menguji konsistensi hasil.
- f. Jika diperlukan, pengujian 75 dan 100 user dapat menjadi penelitian lanjutan untuk menentukan kapasitas maksimum secara sistematis.

5. KESIMPULAN DAN SARAN

Penelitian ini menganalisis kinerja aplikasi Tring Pegadaian menggunakan Apache JMeter pada tiga skenario, yaitu 10, 25, dan 50 pengguna. Hasil pengujian menunjukkan average response time masing-masing sebesar 842 ms, 1.126 ms, dan 1.684 ms, sedangkan throughput masing-masing sebesar 9,8; 21,7; dan 28,9 request/s. Error rate sebesar 0% pada 10 dan 25 user menunjukkan bahwa tidak ada request yang gagal pada kedua skenario tersebut. Pada 50 user, error rate meningkat menjadi 0,40%, setara dengan 2 request gagal dari 500 request, dan success rate sebesar 99,60%.

Analisis persentil juga menunjukkan peningkatan latensi. Dari 10 ke 50 user, median meningkat 93,88%, P90 121,09%, P95 117,45%, dan P99 126,32%. Dengan demikian, peningkatan beban memengaruhi waktu respons, terutama pada bagian ekor distribusi. Skenario 50 User menunjukkan indikasi degradasi dan perlu mendapat perhatian lebih lanjut. Penelitian selanjutnya disarankan untuk melakukan pengujian berulang serta menggabungkan hasil JMeter dengan monitoring CPU, RAM, database, jaringan, dan log aplikasi agar penyebab degradasi dapat diidentifikasi secara lebih spesifik.

DAFTAR REFERENSI

- Abda'u, P. D., Susanto, A., Supriyono, A. R., & Prasetyanti, D. N. (2024). Perbandingan kinerja antara Gatling dan Apache JMeter pada uji beban RESTful API. *Infotekmesin*, 15(1), 211–215. <https://doi.org/10.35970/infotekmesin.v15i1.2176>
- Alnuhait, H., Alzyadat, W., Althunibat, A., Kahtan, H., Zaqaibeh, B., & Al-Khawaja, H. A. (2024). Web application performance assessment: A study of responsiveness, throughput, and scalability. *International Journal of Advanced and Applied Sciences*, 11(9), 214–226. <https://doi.org/10.21833/ijaas.2024.09.023>
- Annisa, A. R., Ardiana, M., Rahayu, P. N., Brian, T., & Jami'in, M. A. (2026). Performance evaluation of web applications using JMeter load testing for server capacity and response efficiency. *Journal of Applied Informatics and Computing*, 10(1), 585–590. <https://doi.org/10.30871/jaic.v10i1.11840>
- Bolanowski, M., Ćmil, M., & Starzec, A. (2024). New model for defining and implementing performance tests. *Future Internet*, 16(10), 366. <https://doi.org/10.3390/fi16100366>
- Cazzola, W., Cesarini, F., & Tansini, L. (2022). PerformERL: A performance testing framework for Erlang. *Distributed Computing*, 35, 439–454. <https://doi.org/10.1007/s00446-022-00429-7>
- Di Meglio, S., Starace, L. L. L., & Di Martino, S. (2026). Semi-automated generation of web app performance tests from end-to-end GUI-level tests with E2E-Loader. *Science of Computer Programming*, 252, 103461. <https://doi.org/10.1016/j.scico.2026.103461>

- Hamidah, I., Haromain, I., & Drehem, I. M. (2025). Evaluasi pengujian kinerja menggunakan JMeter untuk menunjang stabilitas aplikasi layanan perbankan pada PT Bank Rakyat Indonesia Tbk. *DBESTI: Journal of Digital Business and Technology Innovation*, 2(1), 114–126. <https://doi.org/10.54914/dbesti.v2i1.1621>
- Hammad, Y., Al-Said Ahmad, A., & Andras, P. (2025). An empirical study on the performance overhead of code instrumentation in containerised microservices. *Journal of Systems and Software*, 230, 112573. <https://doi.org/10.1016/j.jss.2025.112573>
- Horváth, M., Sakhnenko, V., & Gurbálv I, F. (2024). Comparison of scalability and performance in microservices and monolithic architectures. *2024 IEEE 17th International Scientific Conference on Informatics*. <https://doi.org/10.1109/Informatics62280.2024.10900892>
- Indrianto, I. (2023). Performance testing on web information system using Apache JMeter and BlazeMeter. *Jurnal Ilmiah Ilmu Terapan Universitas Jambi*, 7(2), 138–149. <https://doi.org/10.22437/jiituj.v7i2.28440>
- Leloudas, P. (2023). *Introduction to software testing: A practical guide to testing, design, automation, and execution*. Apress. <https://doi.org/10.1007/978-1-4842-9514-4>
- Mahajan, G., Attar, V., & Kalamkar, S. (2022). Generation of JMeter scripts for performance testing of Moodle server. *2022 4th International Conference on Advances in Computing, Communication Control and Networking (ICAC3N)*. <https://doi.org/10.1109/ICAC3N56670.2022.10074284>
- Puspita, M. A., Mardiana, A. P., R., & Muhammad, M. A. (2025). User acceptance testing dan performance testing pada pengembangan website Kanal Pengetahuan Dikti. *JIEET: Journal of Information Engineering and Educational Technology*, 8(2), 96–101. <https://doi.org/10.26740/jieet.v8n2.p96-101>
- Rachman, A. N., Shofa, R. N., Sjamsuddin, I. N., Tarempa, G. N., Julianto, I. T., & Athoillah, B. A. (2026). Load testing-based performance evaluation of the SiUKT API system. *Journal of Intelligent Systems Technology and Informatics*, 2(1), 22–29. <https://doi.org/10.64878/jistics.v2i1.89>
- Raweyai, S. S., & Widiyari, I. R. (2024). Performance testing of academic website using load testing method supported by Apache JMeter(TM) at XYZ University. *Jurnal Teknik Informatika (JUTIF)*, 5(3), 721–730. <https://doi.org/10.52436/1.jutif.2024.5.3.1796>
- Rotem, A. (2025). *Software performance engineering: A comprehensive guide for high-performance development*. BPB Publications.
- Siahaan, M., & Wijaya, R. (2024). Performance comparison between Laravel and ExpressJs framework using Apache JMeter. *JITE: Journal of Informatics and Telecommunication Engineering*, 7(2), 545–554. <https://doi.org/10.31289/jite.v7i2.10571>