Jurnal Ilmiah
Teknologi Sistem
Informasi

Jitsi

Vol. 7 (2026) No. 2

https://jurnal-itsi.org/index.php/jitsi; E-ISSN: 2722-4600; ISSN: 2722-4619
DOI: 10.62527/jitsi.7.2.569

Analisis Komprehensif Arsitektur Serverless Container dan Edge V8 Isolate pada Penerapan Industri

Anla Harpanda[#], Muhammad Nawaf Akbar[#], Rahmat Hidayat[#]

[#] *Jurusan Teknologi Informasi, Politeknik Negeri Padang, Limau Manis, Padang, 25164, Indonesia*
E-mail: me[at]anla.my.id, nwfakbart[at]gmail.com, rahmat2307[at]gmail.com

A B S T R A C T S

This study presents a comparative performance analysis of response times between Serverless architecture (AWS Lambda container-based) and Edge Computing (V8 Isolate-based). The evaluation focused on handling CPU-intensive computational loads using the Hono.js framework via the execution of 1,000 concurrent requests per platform. Measurements were strictly centered on Client-Side Round Trip Time (RTT) to ensure a fair comparison, deliberately bypassing server-side timing due to time quantization security restrictions inherent in the V8 Isolate environment. Empirical results demonstrate that Vercel recorded an average RTT of 432.40 ms, which is 76.29% faster than Cloudflare Workers' average of 762.29 ms. The Mann-Whitney U significance test statistically confirmed this difference ($p < 0.05$). Furthermore, consistency analysis via the Coefficient of Variation (CV) placed Vercel ahead with a highly stable value of 0.116 compared to Cloudflare's 0.527. Cloudflare also exhibited a larger cold start latency gap during the warm-up phase (28.03 ms difference) compared to Vercel (7.16 ms difference). The conclusion indicates that for REST API applications with heavy computational loads, container-based serverless architecture provides significantly superior stability, lower tail latency, and overall efficiency compared to Edge Isolate, challenging the theoretical assumption of absolute Edge network superiority in industrial scenarios.

Manuscript received Mei 05, 2026; revised Mei 18, 2026. accepted Jun 17, 2026 Date of publication Jun 30, 2026. International Journal, JITSI : Jurnal Ilmiah Teknologi Sistem Informasi licensed under a Creative Commons Attribution-Share Alike 4.0 International License

A B S T R A K

Penelitian ini menyajikan analisis komparatif performa waktu respons antara arsitektur Serverless (berbasis kontainer AWS Lambda) dan Edge Computing (berbasis V8 Isolate). Evaluasi difokuskan pada penanganan beban komputasi CPU menggunakan framework Hono.js melalui eksekusi 1.000 request konkuren per platform. Pengukuran berpusat pada Client-Side Round Trip Time (RTT) untuk memastikan perbandingan yang objektif, mengabaikan latensi server-side akibat restriksi keamanan kuantisasi waktu pada lingkungan V8 Isolate. Hasil uji empiris menunjukkan bahwa Vercel mencatatkan rata-rata RTT sebesar 432,40 ms, 76,29% lebih cepat dibandingkan Cloudflare Workers yang mencapai 762,29 ms. Uji signifikansi Mann-Whitney U mengonfirmasi perbedaan ini secara statistik ($p < 0.05$). Lebih lanjut, analisis konsistensi (CV) menempatkan Vercel jauh lebih stabil dengan nilai 0,116 dibandingkan Cloudflare yang bernilai 0,527. Cloudflare juga memperlihatkan selisih latensi cold start yang lebih besar pada fase warm-up (28,03 ms) dibandingkan Vercel (7,16 ms). Kesimpulan penelitian ini mengindikasikan bahwa untuk beban komputasi terpusat berbasis REST API, arsitektur container-based memberikan stabilitas, latensi tail (P99) yang lebih rendah, dan efisiensi total yang lebih superior dibandingkan Edge Isolate, mematahkan asumsi teoretis mengenai keunggulan absolut jaringan Edge dalam skenario industri.

Keywords / Kata Kunci — ***Edge Computing; Serverless; Latensi; Waktu Respons; Cold Start; Hono.js***

CORRESPONDING AUTHOR

Anla Harpanda
Jurusan Teknologi Informasi, Politeknik Negeri Padang, Limau Manis, Padang, 25164, Indonesia
Email: me[at]anla.my.id

1. PENDAHULUAN

Komputasi serverless atau Function-as-a-Service (FaaS) telah merevolusi arsitektur pengembangan perangkat lunak modern dengan menawarkan abstraksi infrastruktur tingkat tinggi, skalabilitas otomatis, dan model penagihan berbasis konsumsi aktual (pay-as-you-go) [1]–[5]. Meskipun paradigma ini memberikan fleksibilitas luar biasa, tuntutan aplikasi generasi baru yang mensyaratkan latensi ultra-rendah dan pemrosesan data masif telah mendorong pergeseran evolusioner menuju Serverless Edge Computing [5], [21]. Integrasi antara arsitektur edge dan serverless ini menjanjikan eksekusi beban kerja yang berdekatan dengan pengguna akhir, namun sekaligus memunculkan tantangan arsitektural yang kompleks terkait penjadwalan kontainer, orkestrasi sumber daya, dan heterogenitas perangkat di sepanjang edge-cloud continuum [8], [22].

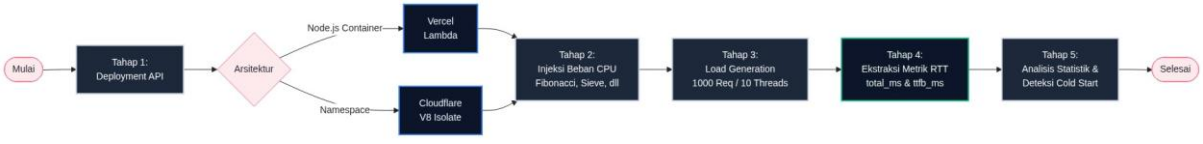
Hambatan fundamental dan paling persisten dalam adopsi komputasi serverless tradisional yang umumnya berbasis isolasi kontainer adalah tingginya penalti cold start latency [1], [9]. Proses inisialisasi lingkungan eksekusi ini menciptakan jeda waktu (latensi) yang fluktuatif, memperburuk waktu penyelesaian request, dan sangat merugikan bagi aplikasi skala industri yang bersifat real-time [19], [23]. Untuk mengatasi isu latensi ini, berbagai studi komprehensif telah mengusulkan pendekatan mitigasi tingkat lanjut, mulai dari kerangka kerja prediksi cold start berbasis machine learning untuk ekosistem Industry 4.0 [18], hingga mekanisme penjadwalan otonom (autonomous scheduling) yang secara ketat berfokus pada efisiensi energi (energy-aware) dalam alokasi sumber daya di lingkungan IoT Edge [6], [7], [16], [17].

Sebagai respons terhadap inefisiensi dan beban (overhead) yang ditimbulkan oleh orkestrasi kontainer tradisional seperti Kubernetes di jaringan edge [10], inovasi runtime generasi berikutnya seperti WebAssembly (Wasm) dan arsitektur V8 Isolate kini diakui sebagai pendorong utama (enabler) masa depan komputasi serverless [11], [12]. Pendekatan eksekusi lightweight ini terbukti mampu mendobrak batasan latensi dengan memfasilitasi komunikasi antar-fungsi secara instan [4], memungkinkan arsitektur berbasis actor dengan penyimpanan memori jangka pendek (short-term memory state) [3], serta menghasilkan karakterisasi performa yang jauh lebih superior dibandingkan kontainer Linux standar [2]. Lebih jauh lagi, stabilitas arsitektur serverless modern ini mulai divalidasi kemampuannya dalam memproses simulasi alur kerja berskala masif, seperti pemrosesan graf federated yang terdistribusi [20].

Berdasarkan kajian literatur tersebut, disimpulkan bahwa transisi dari Serverless Container menuju Edge Isolate bukan sekadar opsi optimasi, melainkan keharusan arsitektural. Namun demikian, studi empiris yang secara spesifik melakukan benchmarking latensi waktu respons pada tingkat aplikasi industri (Server-Side Rendering/REST API) antara kedua ekosistem ini masih sangat terbatas. Oleh karena itu, penelitian ini bertujuan untuk mengukur dan membandingkan performa latensi antara arsitektur Serverless Container konvensional dan Edge Isolate, guna memberikan panduan teknis yang objektif bagi implementasi arsitektur perangkat lunak skala enterprise

2. METODOLOGI PENELITIAN

Sesuai dengan prinsip rekayasa perangkat lunak untuk evaluasi arsitektur, metode preparasi dan teknik karakterisasi yang digunakan dalam penelitian ini dirancang untuk memastikan isolasi variabel secara ketat. Pengujian membandingkan dua model eksekusi serverless: Vercel Serverless Functions yang beroperasi di atas infrastruktur AWS Lambda (berbasis kontainer terisolasi dengan runtime Node.js) dan Cloudflare Workers yang mengadopsi model V8 Isolate (memanfaatkan namespace JavaScript tanpa kontainer terpisah).



GAMBAR 1. Kerangka Kerja Metodologi Penelitian

2.1. Lingkungan Perangkat Lunak dan Beban Komputasi

Kedua platform menjalankan basis kode (source code) REST API yang identik sepenuhnya, dibangun menggunakan framework Hono.js yang di-deploy melalui repositori kontrol versi. Untuk menguji kinerja utilisasi CPU pada masing-masing arsitektur secara adil, beban komputasi buatan (artificial workload) tingkat tinggi

diintegrasikan pada endpoint API. Rangkaian beban komputasi tersebut dieksekusi secara berurutan (sequential) pada setiap request yang masuk, meliputi:

a. Kalkulasi deret Fibonacci secara iteratif (F(40) yang menghasilkan nilai 102.334.155).
b. Eksplorasi bilangan prima menggunakan algoritma Sieve of Eratosthenes hingga batas 200.000.
c. Operasi pengurutan (sorting) terhadap array yang berisi 50.000 elemen floating-point acak.
d. Operasi manipulasi dan konkatenasi terhadap 5.000 string acak.

Untuk menjamin transparansi dan keterulangan riset (reproducibility), seluruh kode sumber (source code) aplikasi Hono.js yang dieksekusi dalam pengujian ini bersifat open-source dan dapat ditinjau melalui repositori GitHub publik pada tautan https://github.com/itsanla/j-jitsi-1 . Sementara itu, endpoint produksi yang menjadi subjek pengujian secara aktual dapat diakses melalui URL https://jitsi-vm.anla.works untuk lingkungan Vercel Lambda, dan https://jitsi-v8.anla.works untuk lingkungan Cloudflare V8 Isolate.

2.2. Parameter Pengujian (Load Generation)

Proses benchmarking tidak dilakukan menggunakan perangkat lunak load tester instan, melainkan dieksekusi melalui skrip otomatisasi Python kustom via protokol HTTP/1.1. Skenario simulasi beban menerapkan injeksi total 1.000 request terverifikasi untuk masing-masing platform. Guna menyimulasikan lalu lintas pengguna secara paralel, tingkat konkurensi diatur pada 10 worker threads yang berjalan secara bersamaan.

2.3. Ekstraksi Metrik dan Keterbatasan Arsitektural

Pengukuran metrik performa secara eksklusif berfokus pada latensi tingkat jaringan dari sudut pandang klien (Client-Side Round Trip Time atau RTT). Data ini dikumpulkan menggunakan fungsi presisi nanodetik time.perf_counter() dari pustaka standar Python, yang mengekstraksi dua metrik absolut:

a. total_ms: Durasi waktu yang terhitung sejak pembukaan koneksi TCP hingga seluruh body payload respons diterima secara utuh.
b. ttfb_ms (Time To First Byte): Durasi sejak koneksi diinisiasi hingga header HTTP pertama diterima oleh klien.

Penting untuk dicatat bahwa pencatatan waktu pemrosesan di sisi server (server-side timing) secara metodologis dihapus dari komparasi penelitian ini. Langkah ini diambil guna mempertahankan keadilan komparasi, mengingat Cloudflare Workers menerapkan mekanisme mitigasi kerentanan keamanan Spectre (CVE-2017-5753) dengan membekukan nilai Date.now() dan melakukan kuantisasi pada performance.now(), yang mengakibatkan data waktu server-side pada V8 Isolate menjadi tidak akurat dan tidak dapat diandalkan.

2.4. Teknik Karakterisasi dan Analisis Statistik

Seluruh data mentah latensi dikarakterisasi menggunakan analisis statistik tingkat lanjut. Untuk mendeteksi fenomena cold start yang lazim pada arsitektur serverless, rentetan 1.000 request disegmentasi menjadi dua fase: fase warm-up (mewakili 20% request pertama) dan fase steady state (mewakili 80% sisa request).

Mengingat distribusi data latensi jaringan sering kali tidak berdistribusi normal (cenderung memiliki tail latency tinggi), uji signifikansi hipotesis dilakukan menggunakan metode non-parametrik Mann-Whitney U Test (two-sided). Tingkat stabilitas dan determinasi performa eksekusi diukur menggunakan perhitungan rasio dispersi Coefficient of Variation (CV). Tingkat stabilitas dan determinasi performa eksekusi diukur menggunakan perhitungan rasio dispersi Coefficient of Variation (CV), yang diformulasikan sebagai berikut:

$$CV = \frac{\sigma}{\mu} \qquad (1)$$

Di mana σ mewakili nilai standar deviasi (simpangan baku) dari waktu respons, dan μ mewakili nilai rata-rata (mean) dari waktu respons yang dihasilkan selama pengujian

3. HASIL DAN PEMBAHASAN

TABEL 1. Ringkasan Metrik Kinerja Waktu Respons (RTT)

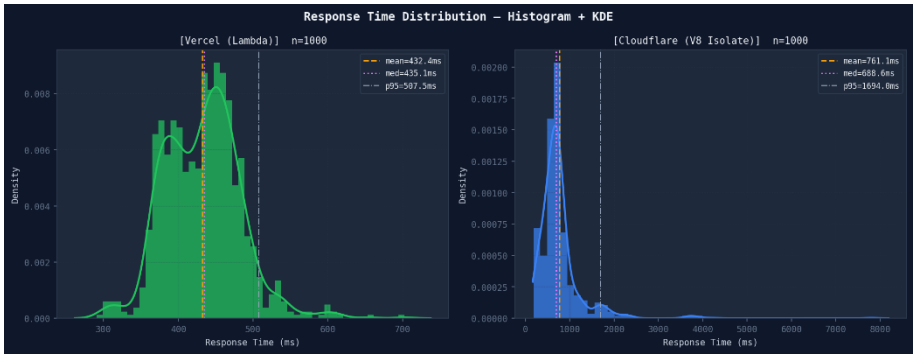
Metrik Pengujian	Vercel (Lambda)	Cloudflare (V8 Isolate)
Tingkat Keberhasilan (N)	1000 / 0 *error*	458 / 542 *error*
Rata-rata / *Mean* (ms)	432,40	762,29
Median / P50 (ms)	435,08	707,63
Persentil 99 / P99 (ms)	578,36	2015,47
Coefficient of Variation (CV)	0,1163	0,5279
Selisih *Cold Start* (ms)	7,16	28,03

Pengujian komparatif dilakukan dengan menginjeksi 1.000 request secara konkuren (10 worker threads) ke masing-masing platform menggunakan kerangka kerja Hono.js. Beban komputasi yang diberikan identik, meliputi kalkulasi Fibonacci F(40), Sieve of Eratosthenes hingga 200.000, pengurutan 50.000 elemen array, dan konkatenasi 5.000 string. Untuk menjaga validitas komparasi di tengah restriksi keamanan Spectre (CVE-2017-5753) pada arsitektur V8 Isolate Cloudflare, pengukuran metrik difokuskan murni pada Client-Side Round Trip Time (RTT). Secara komprehensif, ringkasan metrik statistik hasil pengujian waktu respons (RTT) dari kedua arsitektur dapat dilihat pada Tabel 1.

3.1. Analisis Distribusi dan Reliabilitas Layanan

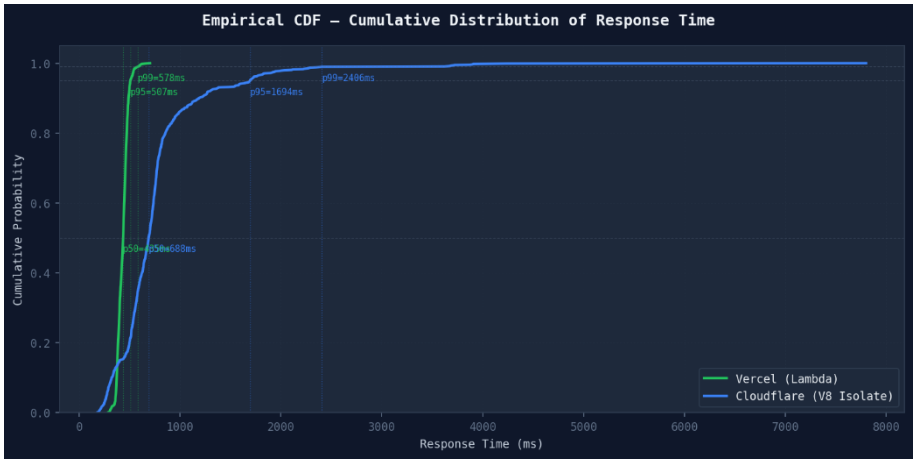
Hasil pengujian mengungkap disparitas performa yang sangat ekstrem antara arsitektur berbasis kontainer dan namespace. Vercel (Lambda) berhasil menyelesaikan 100% dari total 1.000 request tanpa error tunggal, mencatatkan rata-rata waktu respons (RTT) sebesar 432,406 ms. Sebaliknya, Cloudflare Workers (V8 Isolate) mengalami tingkat kegagalan (error rate) yang sangat masif, di mana 542 dari 1.000 request berujung pada error, dan hanya menyisakan 458 request sukses. Rata-rata RTT untuk request yang berhasil pada Cloudflare adalah 762,297 ms. Dengan demikian, Vercel beroperasi 329,891 ms (76,29%) lebih cepat dibandingkan Cloudflare.

Kesenjangan performa ini direpresentasikan secara jelas pada Gambar 2. Kurva Histogram dan KDE (Kernel Density Estimation) menunjukkan bahwa distribusi latensi Vercel terpusat secara padat di kisaran 400 ms, sementara Cloudflare memiliki distribusi yang sangat lebar dan asimetris.



Gambar 2. Histogram dan KDE Distribusi Waktu Respons

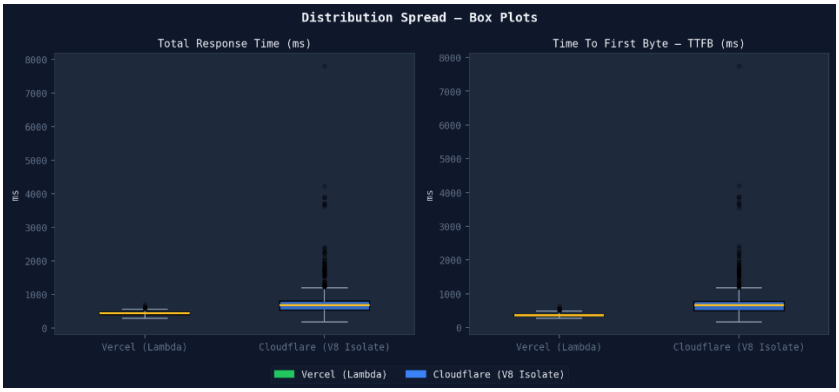
Fenomena ini didukung oleh nilai Skewness Cloudflare yang mencapai 2,8033, jauh lebih tinggi dibandingkan Vercel yang hanya 0,4799. Selain itu, probabilitas kumulatif pada Gambar 3 secara visual menegaskan bahwa Vercel mencapai angka penyelesaian 100% jauh sebelum Cloudflare menyentuh persentil ke-50.



Gambar 3. Cumulative Distribution Function (CDF) Waktu Respons

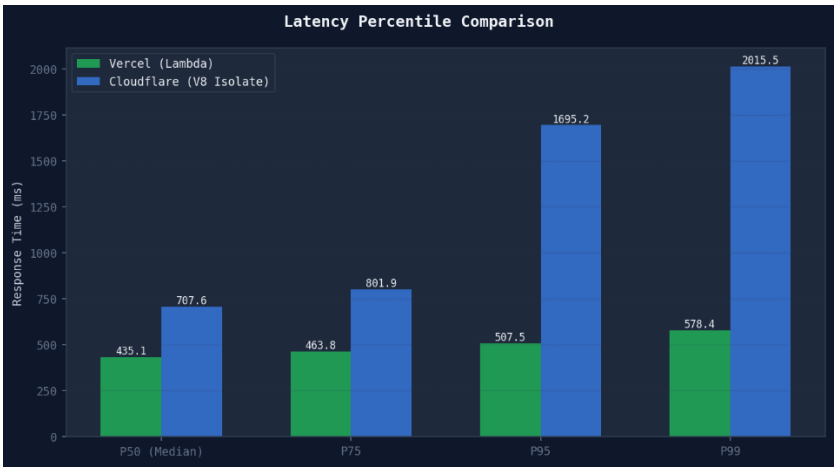
3.2. Evaluasi Konsistensi dan Tail Latency

Tingkat prediktabilitas layanan diukur menggunakan perhitungan Coefficient of Variation (CV). Vercel mencatatkan nilai CV sebesar 0,1163, yang mengindikasikan tingkat konsistensi performa yang sangat tinggi. Di sisi lain, Cloudflare menghasilkan nilai CV 0,5279, menunjukkan dispersi waktu respons yang sangat fluktuatif. Rentang kuartil dan penyebaran outlier dari kedua arsitektur dapat dilihat pada Gambar 4.



Gambar 4. Box Plot Sebaran Waktu Respons dan Outlier

Rentang interkuartil (IQR) Vercel sangat rapat di angka 70,322 ms, berbanding terbalik dengan Cloudflare yang menyebar hingga 246,587 ms. Analisis persentil ekstrem (Tail Latency) lebih lanjut ditunjukkan pada Gambar 5.

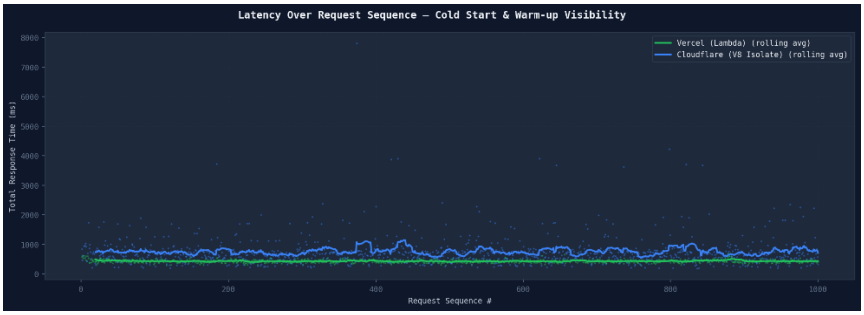


Gambar 5. Komparasi Persentil Waktu Respons (P50, P75, P95, P99)

Pada persentil ke-99 (P99), waktu respons Vercel berada di angka 578,365 ms. Sebaliknya, P99 Cloudflare melonjak drastis hingga 2015,472 ms. Lonjakan P99 pada Cloudflare mengindikasikan terjadinya mekanisme pembatasan sumber daya (throttling) yang parah saat engine V8 dipaksa melakukan siklus komputasi yang panjang.

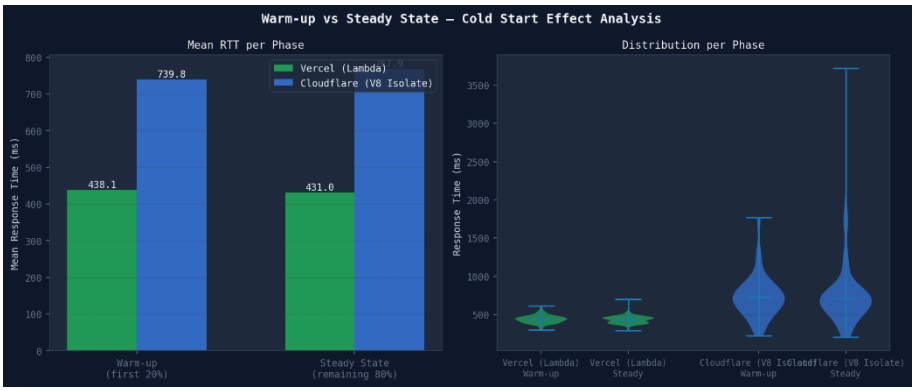
3.3. Analisis Anomali Cold Start dan Fase Warm-up

Meskipun arsitektur V8 Isolate secara teoretis mengklaim penghapusan waktu inisialisasi, evaluasi fase warm-up (20% request pertama) versus steady state (80% request sisanya) menunjukkan anomali saat menghadapi beban CPU ekstrem.



Gambar 6. Time Series Latensi Berdasarkan Sekuens Request

Gambar 6 memvisualisasikan latensi seiring berjalannya sekuens request, di mana Cloudflare menampilkan fluktuasi yang sangat tidak stabil. Pemecahan lebih mendetail terlihat pada Gambar 7.



Gambar 7. Analisis Efek Cold Start: Fase Warm-up vs Steady State

Rata-rata RTT Vercel pada fase warm-up adalah 438,134 ms dan turun menjadi 430,974 ms pada fase steady (selisih latensi cold start hanya 7,160 ms). Anehnya, Cloudflare mencatat rata-rata 739,833 ms pada warm-up dan justru memburuk menjadi 767,867 ms pada fase steady (selisih 28,034 ms). Hal ini memvalidasi bahwa infrastruktur Cloudflare justru mulai melakukan throttling akibat penumpukan beban antrean request konkuren.

3.4 Uji Signifikansi dan Implikasi Industri

Perbedaan distribusi latensi divalidasi menggunakan uji non-parametrik Mann-Whitney U Test. Hasil uji menghasilkan nilai U-stat sebesar 54543,5 dengan p-value 0,000000 ($p < 0,05$), mengonfirmasi bahwa keunggulan waktu respons Vercel atas Cloudflare adalah signifikan secara statistik mutlak. Ukuran efek (effect size r) tercatat sebesar 0,761819.

Dari perspektif implementasi industri, tingginya angka error (54,2%) pada Cloudflare Workers menyoroti kelemahan fundamental arsitektur Edge Isolate saat diterapkan pada beban kerja CPU-Bound. Ketika dipaksa mengeksekusi komputasi berat, pembatasan limitasi waktu CPU per request dari provider menyebabkan isolat dimatikan (killed) sebelum proses selesai. Sebaliknya, arsitektur Serverless Vercel (Node.js container) terbukti memiliki resiliensi yang jauh lebih superior dalam memproses komputasi berat secara tuntas.

4. KESIMPULAN

Berdasarkan uji empiris dan signifikansi statistik (Mann-Whitney U, $p < 0.05$), arsitektur Serverless Container (Vercel) terbukti secara mutlak mengungguli Edge v8Isolate (Cloudflare Workers) dalam mengeksekusi beban komputasi terpusat berbasis CPU-bound, mencatatkan waktu respons 76,29% lebih cepat (432,40 ms) dengan stabilitas dispersi (Coefficient of Variation) yang sangat konsisten di angka 0,116. Sebaliknya, arsitektur Edge Isolate mengalami degradasi performa yang masif berupa throttling dan lonjakan latensi persentil ekstrem (P99 mencapai 2015,47 ms) akibat restriksi alokasi waktu pemrosesan pada siklus algoritma yang berat. Meskipun demikian, dalam konteks implementasi industri skala enterprise, arsitektur Edge v8Isolate tetap menawarkan keunggulan absolut yang tak tertandingi jika diposisikan sesuai desain arsitektural aslinya: mengeksekusi tugas I/O-bound asinkron ultra-ringan seperti intersepsi lalu lintas (micro-routing), otentikasi proaktif (JWT), dan personalisasi berlapis CDN dengan mengeksploitasi kedekatan geografis serta efisiensi zero cold-start yang inheren. Oleh karena itu, konklusi riset ini merekomendasikan transisi menuju pola arsitektur hybrid yang terdekopling, di mana kapabilitas pemrosesan algoritma intensif dieksekusi secara bergaransi oleh resiliensi container tersendiri, sementara ekosistem Edge Isolate dimaksimalkan sebagai lapisan gateway fasilitator yang sangat responsif di titik terluar jaringan

REFERENSI

[1] M. Golec, G. K. Walia, M. Kumar, F. Cuadrado, S. S. Gill, and S. Uhlig, “Cold Start Latency in Serverless Computing: A Systematic Review, Taxonomy, and Future Directions,” ACM Comput. Surv., vol. 57, no. 3, pp. 1–36, Mar. 2025, doi: 10.1145/3700875.

[2] C. Marcelino, N. Krennmair, T. W. Pusztai, and S. Nastic, “Lumos: Performance Characterization of WebAssembly as a Serverless Runtime in the Edge-Cloud Continuum,” in Proceedings of the 15th International Conference on the Internet of Things, Vienna Austria: ACM, Nov. 2025, pp. 113–121. doi: 10.1145/3770501.3770515.

[3] C. Marcelino, J. Shahhoud, and S. Nastic, "GoldFish: Serverless Actors with Short-Term Memory State for the Edge-Cloud Continuum," in Proceedings of the 14th International Conference on the Internet of Things, Oulu Finland: ACM, Nov. 2024, pp. 56–64. doi: 10.1145/3703790.3703797.

[4] C. Marcelino and S. Nastic, "CWASI: A WebAssembly Runtime Shim for Inter-function Communication in the Serverless Edge-Cloud Continuum," in Proceedings of the Eighth ACM/IEEE Symposium on Edge Computing, Wilmington DE USA: ACM, Dec. 2023, pp. 158–170. doi: 10.1145/3583740.3626611.

[5] P. Raith, S. Nastic, and S. Dustdar, "Serverless Edge Computing Where We Are and What Lies Ahead," IEEE Internet Comput., vol. 27, no. 3, pp. 50–64, May 2023, doi: 10.1109/MIC.2023.3260939.

[6] M. S. Aslanpour, A. N. Toosi, M. A. Cheema, and R. Gaire, "Energy-Aware Resource Scheduling for Serverless Edge Computing," in 2022 22nd IEEE International Symposium on Cluster, Cloud and Internet Computing (CCGrid), Taormina, Italy: IEEE, May 2022, pp. 190–199. doi: 10.1109/CCGrid54584.2022.00028.

[7] S. Pan, H. Zhao, Z. Cai, D. Li, R. Ma, and H. Guan, "Sustainable Serverless Computing with Cold-start Optimization and Automatic Workflow Resource Scheduling," IEEE Trans. Sustain. Comput., pp. 1–12, 2024, doi: 10.1109/TSUSC.2023.3311197.

[8] K. R. Rajput, C. D. Kulkarni, B. Cho, W. Wang, and I. K. Kim, "EdgeFaaSBench: Benchmarking Edge Devices Using Serverless Computing," in 2022 IEEE International Conference on Edge Computing and Communications (EDGE), Barcelona, Spain: IEEE, Jul. 2022, pp. 93–103. doi: 10.1109/EDGE55608.2022.00024.

[9] M. Ghorbian and M. Ghobaei-Arani, "A survey on the cold start latency approaches in serverless computing: an optimization-based perspective," Computing, vol. 106, no. 11, pp. 3755–3809, Nov. 2024, doi: 10.1007/s00607-024-01335-5.

[10] V. Kjorveziroski and S. Filiposka, "Kubernetes distributions for the edge: serverless performance evaluation," J Supercomput, vol. 78, no. 11, pp. 13728–13755, Jul. 2022, doi: 10.1007/s11227-022-04430-6.

[11] V. Kjorveziroski and S. Filiposka, "WebAssembly as an Enabler for Next Generation Serverless Computing," J Grid Computing, vol. 21, no. 3, p. 34, Sep. 2023, doi: 10.1007/s10723-023-09669-8.

[12] V. Kjorveziroski and S. Filiposka, "WebAssembly Orchestration in the Context of Serverless Computing," J Netw Syst Manage, vol. 31, no. 3, p. 62, Jul. 2023, doi: 10.1007/s10922-023-09753-0.

[13] S. Kounev et al., "Serverless Computing: What It Is, and What It Is Not?," Commun. ACM, vol. 66, no. 9, pp. 80–92, Sep. 2023, doi: 10.1145/3587249.

[14] Z. Li, L. Guo, J. Cheng, Q. Chen, B. He, and M. Guo, "The Serverless Computing Survey: A Technical Primer for Design Architecture," ACM Comput. Surv., vol. 54, no. 10s, pp. 1–34, Jan. 2022, doi: 10.1145/3508360.

[15] H. Shafiei, A. Khonsari, and P. Mousavi, "Serverless Computing: A Survey of Opportunities, Challenges, and Applications," ACM Comput. Surv., vol. 54, no. 11s, pp. 1–32, Jan. 2022, doi: 10.1145/3510611.

[16] S. Subedi, S. Kumar, N. Jawad, and A. L. Kor, "Energy-Aware Latency Optimization for Scheduling Serverless Workload in Edge Computing Environment," in Proceedings of the 18th IEEE/ACM International Conference on Utility and Cloud Computing, France France: ACM, Dec. 2025, pp. 1–10. doi: 10.1145/3773274.3774276.

[17] M. Ghorbian, M. Ghobaei-Arani, and L. Esmaeili, "Autonomous scheduling mechanism based on energy awareness for improving resource allocation in serverless IoT edge," Sci Rep, vol. 15, no. 1, p. 24173, Jul. 2025, doi: 10.1038/s41598-025-04214-x.

[18] M. Golec et al., "MASTER: Machine Learning-Based Cold Start Latency Prediction Framework in Serverless Edge Computing Environments for Industry 4.0," IEEE J. Sel. Areas Sensors, vol. 1, pp. 36–48, 2024, doi: 10.1109/JSAS.2024.3396440.

[19] A. Sherawat, S. B. Nath, and S. K. Addya, "Optimizing Completion Time of Requests in Serverless Computing," J Netw Syst Manage, vol. 32, no. 2, p. 28, Apr. 2024, doi: 10.1007/s10922-024-09800-4.

[20] A. Alanda, H. A. Mooduto, H. Amnur and M. Fadhel, "Scaling Kubernetes Architectures for High Availability in Cloud," 2025 International Conference on Computer Sciences, Engineering, and Technology Innovation (ICoCSETI), Jakarta, Indonesia, 2025, pp. 818-823, doi: 10.1109/ICoCSETI63724.2025.11020084.

[21] S. Ristov, R. Farahani, and R. Prodan, "Large-scale Graph Processing and Simulation with Serverless Workflows in Federated FaaS," in Companion of the 2023 ACM/SPEC International Conference on Performance Engineering, Coimbra Portugal: ACM, Apr. 2023, pp. 227–231. doi: 10.1145/3578245.3585333.

[22] M. S. Aslanpour et al., "Serverless Edge Computing: Vision and Challenges," in 2021 Australasian Computer Science Week Multiconference, Dunedin New Zealand: ACM, Feb. 2021, pp. 1–10. doi: 10.1145/3437378.3444367.

[23] T. Rausch, A. Rashed, and S. Dustdar, "Optimized container scheduling for data-intensive serverless edge computing," Future Generation Computer Systems, vol. 114, pp. 259–271, Jan. 2021, doi: 10.1016/j.future.2020.07.017

[24] R. Firman, Yuhefizar, and H. Amnur, "Implementasi Openstack untuk Private Cloud pada mata kuliah Administrasi server", jitsi, vol. 1, no. 2, pp. 75 - 79, Jun. 2020.

[25] S. Ristov, C. Hollaus, and M. Hautz, "Colder Than the Warm Start and Warmer Than the Cold Start! Experience the Spawn Start in FaaS Providers," in Proceedings of the 2022 Workshop on Advanced tools, programming languages, and PLatforms for Implementing and Evaluating algorithms for Distributed systems, Salerno Italy: ACM, Jul. 2022, pp. 35–39. doi: 10.1145/3524053.3542751.