IJEEC

International Journal of Electrical Engineering and Computing
Vol. 10, No. 1 (2026)

Original research paper
UDC
DOI

End-to-End Latency Decomposition in AI-Driven Web Applications: Rethinking Infrastructure in LLM-Based Systems

Jasna Hamzabegović, Amel Džanić, Kenan Duraković

Faculty of Technical Engineering, University of Bihać, Bihać, Bosnia and Herzegovina

E-mail address: jasna.hamzabegovic@unbi.ba, amel.dzanic@unbi.ba, kenandurakovic.tm87@gmail.com

Abstract—**The increasing integration of artificial intelligence into web applications, particularly through large language models (LLMs), has fundamentally reshaped the performance characteristics of modern systems. Unlike traditional architectures, where latency is primarily determined by backend infrastructure, AI-driven applications operate as multi-stage pipelines involving orchestration logic, network communication, and external model inference. This paper introduces an end-to-end latency decomposition framework for analyzing performance in AI-powered web applications. A controlled experimental study is conducted using two production-equivalent implementations deployed in serverless and virtual private server (VPS) environments. The methodology distinguishes between full-stack execution, including LLM inference, and infrastructure-only scenarios, enabling precise isolation of latency contributions across infrastructure, application, and model layers. The results indicate that in full-stack scenarios, model-related latency dominates system performance, accounting for approximately 85% of total response time, thereby minimizing the impact of infrastructure differences. In contrast, infrastructure-only scenarios reveal significant performance variations between deployment environments. These findings challenge infrastructure-centric optimization approaches and demonstrate the need for system-level performance evaluation in LLM-based applications. The proposed framework provides a practical methodology for identifying performance bottlenecks and offers actionable insights for optimizing AI-driven web systems.**

Keywords-AI web applications; latency decomposition; large language models (LLM): serverless computing; virtual private server (VPS); end-to-end latency; performance evaluation; cloud computing; AI systems; benchmarking

I. INTRODUCTION

The rapid integration of artificial intelligence into web-based systems has fundamentally reshaped the performance landscape of modern applications. In particular, the adoption of large language models as externally hosted services has introduced a new layer of complexity, where system responsiveness is no longer determined solely by backend infrastructure, but increasingly by multi-stage processing pipelines involving network communication, orchestration logic, and model inference [1], [2], [3].

Traditional performance analysis in web systems has predominantly focused on infrastructure-level optimization, comparing deployment paradigms such as serverless architectures and virtual private servers in terms of latency, scalability, resource utilization, and cost efficiency [4], [5], [6]. These studies typically rely on empirical benchmarking approaches, where performance metrics such as response time, throughput, and pricing models are evaluated across different cloud deployment strategies to determine the most suitable architecture for specific workloads [7], [8].

However, this assumption becomes increasingly questionable in the context of modern AI-driven applications, where large language models are frequently consumed as external services and integrated into multi-stage processing pipelines [1], [2], [3].

Despite these architectural shifts, existing performance evaluation approaches have not fully adapted to this new paradigm. Most studies continue to emphasize infrastructure comparison, while the relative contribution of different components within AI application pipelines remains insufficiently explored, particularly in real-world deployment scenarios [7], [9].

This paper addresses this gap by introducing an end-to-end latency decomposition approach for AI-powered web applications. The study is motivated by the observation that performance bottlenecks may shift away from infrastructure toward model-centric components, challenging conventional assumptions about optimization strategies. Instead of focusing solely on comparative infrastructure benchmarking, this work aims to quantify the relative contribution of different layers within the application pipeline.

To achieve this, a controlled experimental setup is employed, based on two production-equivalent implementations of the same application deployed in distinct

infrastructural environments. The key methodological innovation lies in separating full-stack execution, which includes external model calls, from infrastructure-only execution, where model interaction is replaced with deterministic responses. This design enables precise isolation of latency sources and allows for a fine-grained analysis of system behavior under varying load conditions.

Preliminary findings indicate that, in full-stack AI scenarios, the dominant portion of response time is attributable to the model layer, while infrastructural differences appear to have a comparatively limited impact. Conversely, when the model component is removed, infrastructure-related latency differences become significant and measurable. These findings suggest that infrastructure optimization alone may not yield substantial performance improvements in AI-driven systems where external services govern the critical path.

The main contribution of this paper is twofold. First, it provides an empirical framework for decomposing latency in AI web applications, enabling a more accurate identification of performance bottlenecks. Second, it reframes the role of infrastructure in modern AI systems, showing that its importance must be evaluated in relation to the broader application pipeline rather than in isolation. By shifting the focus from infrastructure-centric to system-level analysis, this work contributes to a more nuanced understanding of performance optimization in contemporary AI-enabled web architectures. This study extends a related line of research on infrastructure performance in AI applications by introducing a latency decomposition perspective that enables system-level analysis beyond comparative benchmarking.

In this context, the central research question of this study is to what extent infrastructure contributes to end-to-end latency in AI-driven web applications when compared to model-related processing. More specifically, the study investigates whether infrastructure-level optimization remains a dominant factor in performance, or whether its impact becomes secondary in the presence of external AI services.

Based on the research question, the following hypotheses are formulated:

H1: In full-stack AI-driven scenarios, model-related latency dominates end-to-end response time, reducing the relative impact of infrastructure differences between serverless and VPS architectures.

H2: When model inference is excluded from the execution path, infrastructure-related latency becomes the dominant contributor to overall system performance, revealing measurable differences between deployment architectures.

II. Related Work

Performance evaluation of cloud-based web systems has traditionally been dominated by infrastructure-centric analyses, with a particular focus on comparing deployment paradigms such as serverless computing, virtual machines, and microservices architectures. Existing studies consistently evaluate performance across dimensions such as latency, throughput, scalability, and cost efficiency, often through controlled benchmarking experiments and workload simulations [10], [11]. These approaches typically assume that infrastructure constitutes the primary determinant of system performance, leading to a strong emphasis on selecting optimal deployment environments.

To support such evaluations, a significant body of research has developed benchmarking methodologies and frameworks tailored to cloud-native systems. Prior work highlights the importance of moving beyond micro-benchmarks toward realistic workload scenarios and end-to-end evaluation strategies [12], [13]. Advanced benchmarking approaches based on distributed tracing and workflow execution demonstrate that performance variability in serverless environments is influenced by orchestration overhead, event triggers, and external service dependencies rather than raw compute performance alone [7].

At the same time, research on serverless computing has identified inherent performance challenges related to cold-start latency, resource isolation, and distributed execution overhead. Empirical studies show that initialization delays and runtime variability can significantly impact response time, particularly in latency-sensitive applications [14], [15]. Additional work on high-performance serverless systems further highlights limitations such as scheduling overhead and network communication inefficiencies, reinforcing the complexity of performance behavior in modern cloud environments [16].

More recently, the emergence of AI-driven web applications has introduced a new dimension to performance analysis. Modern systems increasingly rely on distributed pipelines that integrate data retrieval, orchestration logic, and external model inference, particularly in architectures based on large language models and retrieval-augmented generation [1], [2]. In such systems, performance is no longer confined to infrastructure boundaries but is instead shaped by interactions between multiple distributed components, including external APIs, data services, and vector databases [17].

Despite these advances, existing research largely treats infrastructure performance and AI pipeline design as separate concerns. While infrastructure-focused studies provide insights into deployment trade-offs, and AI-focused research explores model integration and data pipelines, there remains a lack of unified approaches that analyze their combined effect on end-to-end latency. In particular, there is limited empirical work that decomposes system latency across infrastructural and model-related components within realistic application scenarios.

This paper addresses this gap by bridging infrastructure-centric benchmarking and AI system analysis through an end-to-end latency decomposition framework. By explicitly separating infrastructure-related and model-related contributions to response time, the proposed approach enables a more accurate identification of performance bottlenecks and offers a revised perspective on the role of infrastructure in modern AI-powered web systems.

Unlike tracing frameworks that focus on fine-grained event visualization and workflow execution, the proposed framework emphasizes system-level attribution of latency contributions across infrastructure, application, and model layers. This

perspective provides additional insight into AI-driven systems, where external model inference dominates the critical execution path.

III. METHODOLOGY

This study adopts a structured experimental approach to analyze performance characteristics of AI-driven web applications, with a particular focus on identifying the relative contribution of infrastructure and model-related components to overall system latency. Unlike traditional benchmarking studies that emphasize total response time, the proposed methodology is designed to enable a fine-grained analysis of latency by decomposing it into its constituent parts. To achieve this, a dedicated latency decomposition framework is introduced and evaluated through controlled experiments across different deployment environments.

A. Latency Decomposition Framework

Modern AI-driven web applications operate as multi-stage distributed systems, where user requests traverse multiple layers before a response is generated. Unlike traditional web architectures, where latency is primarily attributed to backend processing and infrastructure, AI-powered systems introduce additional components such as external model inference, orchestration logic, and data retrieval pipelines. As a result, total response time must be understood as a composition of multiple interacting factors rather than a single infrastructure-dependent metric.

In this study, system latency is modeled as a composite function of distinct processing layers. The total end-to-end latency can be conceptually decomposed into three primary components: infrastructure latency, application-level processing latency, and model-related latency.

$$T_{total} = T_{infra} + T_{app} + T_{model} \quad (1)$$

As shown in (1), this decomposition enables a structured analysis of performance bottlenecks by isolating the contribution of each component. In contrast to traditional benchmarking approaches, which primarily evaluate total response time, this framework explicitly separates model-dependent and infrastructure-dependent factors, allowing for a more precise interpretation of system behavior.

The proposed latency decomposition framework is illustrated in Fig. 1.

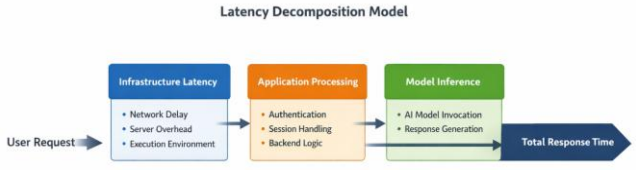

Fig. 1. Latency decomposition model for AI-driven web applications.

B. Experimental Design and Reproducibility

To operationalize the proposed framework, a controlled experimental setup was designed using two production-equivalent implementations of the same AI web application. The first implementation was deployed in a serverless environment, while the second was deployed on a virtual private server. Both versions shared identical application logic, database configuration, geographic deployment region, and external AI service integration, ensuring that observed differences could be attributed to infrastructural characteristics rather than functional discrepancies.

A key methodological element of this study is the separation of execution modes into two distinct scenarios:

- Full-stack scenario, in which the complete processing pipeline is executed, including external model invocation
- Infrastructure-only scenario, in which the model layer is replaced by a deterministic response, enabling isolation of non-model latency

This dual-scenario design allows direct comparison between total latency and infrastructure-dominated latency, providing empirical support for latency decomposition.

The VPS environment was hosted on a Hetzner CX22 instance (2 vCPU, 4 GB RAM) located in the European region. External inference was performed using OpenAI GPT-4o through the same API endpoint.

C. Experimental Environment

To ensure fair comparison and reproducibility, two production-equivalent implementations of the same AI-driven web application were evaluated. The first implementation was deployed using Vercel serverless functions, whereas the second was hosted on a Hetzner VPS infrastructure with persistent execution. Both deployments shared identical application logic, API routes, database configuration, user workflows, authentication mechanisms, and geographical execution region.

The application was implemented using the Next.js framework with API routes and a chat-based user interface. Two endpoints were instrumented for experimental purposes. The /api/chat endpoint represented the complete processing pipeline, including user authentication, session validation, context preparation, and interaction with an external large language model. The /api/chat-mock endpoint replaced model inference with a deterministic response while preserving application-level operations, enabling isolation of infrastructure-related latency. External model inference was performed through the OpenAI API using the same model configuration, prompt structure, and request parameters across both deployment environments. This ensured that any observed differences in end-to-end latency originated from architectural characteristics rather than variations in model behavior.

Performance evaluation was conducted using the k6 load-testing framework. Three workload conditions were considered. The cold scenario consisted of a single request executed after an idle period of at least 15 minutes. The warm scenario consisted of thirty consecutive requests generated by one virtual user. The load scenario employed a staged increase of concurrent users from 10 to 25 and finally 50 virtual users.

To support latency decomposition, custom server-side instrumentation was introduced through HTTP response headers. Total server latency (x-total-ms), model inference time (x-llm-ms), and application processing time (x-app-ms) were recorded for each request and exported through custom k6 trend metrics. Experimental results were automatically stored in structured JSON and Markdown artifacts to ensure traceability and reproducibility. All experiments were executed under identical network conditions using the same client machine and standardized environment variables.

Model latency measurements included external API communication and provider-side inference time as observed from the application layer, encompassing network transmission, model processing, token generation, and response serialization.

D. Workload Modeling and Test Scenarios

System performance was evaluated under three representative workload conditions designed to reflect realistic application behavior:

- Cold scenario: a single request executed after a prolonged idle period, capturing initialization overhead and cold-start latency
- Warm scenario: a sequence of consecutive requests with a single user, representing steady-state operation
- Load scenario: a gradually increasing number of concurrent users, simulating real-world demand and scalability conditions

All experiments were conducted using standardized load-testing tools, with identical request structures, authentication flows, and session contexts across both deployment environments. The same set of input prompts and interaction patterns was used to ensure consistency between test runs.

E. Measurement and Instrumentation

To enable fine-grained latency analysis, the system was instrumented to capture both client-side and server-side performance metrics. Standard HTTP metrics such as total request duration and error rate were collected, alongside custom server-side measurements designed to reflect the proposed decomposition model.

Specifically, response headers were extended to include:

- total processing time
- model execution time
- application-level processing time

These measurements were mapped to the components of the latency decomposition model, allowing the total response time to be partitioned into model-related and non-model contributions. Additional derived metrics were computed to analyze variability, including median latency and tail distributions under load conditions.

F. Reproducibility and Control Variables

To ensure experimental validity and reproducibility, all tests were conducted under controlled conditions. The same client machine, network environment, and execution schedule were used for all experiments. Idle intervals were enforced between cold-start tests, and workload sequences were executed in a fixed order to minimize external variability.

All configuration parameters, including API endpoints, authentication tokens, and test identifiers, were standardized and documented. Results were automatically recorded and exported in structured formats, enabling traceability and repeatability of experimental runs.

G. Metric Definitions

To ensure consistency and interpretability of performance evaluation, key metrics are formally defined. Median latency is used as a robust measure of central tendency, providing a stable estimate of typical system response time under varying conditions. In addition, p95 latency is employed to capture tail behavior, reflecting worst-case performance experienced by users under load.

Error rate is used as an indicator of system reliability, particularly under increasing workload conditions. Furthermore, decomposition metrics ($T_{(infra)}$, $T_{(app)}$, $T_{(model)}$) are derived from server-side instrumentation and mapped to the proposed latency decomposition framework. These metrics enable a quantitative assessment of the relative contribution of individual system components to total response time.

H. Statistical Analysis

Latency values reported in this study were derived from repeated benchmark executions conducted under identical experimental conditions. Performance metrics were automatically computed by the k6 load-testing framework, including average (avg), median (med), p90, p95, and p99 latency statistics. Median latency was selected as the primary measure of central tendency because it is less sensitive to outliers and transient variations than the arithmetic mean. Tail behavior was primarily characterized using p95 latency values, while p99 statistics were additionally inspected to verify the consistency of latency behavior under higher-percentile conditions.

Since the objective of this work is comparative rather than inferential, no formal significance tests were performed. The analysis focuses on relative differences between architectures and the decomposition of latency contributions across infrastructure, application, and model layers.

Workload scenarios were designed to ensure consistency and comparability across experiments. Warm scenarios consisted of 30 consecutive requests generated by a single virtual user, whereas load scenarios employed staged concurrency levels of 10, 25, and 50 virtual users. Cold scenarios were executed after an idle period of at least 15 minutes to capture initialization effects and cold-start behavior. All measurements were collected under identical network conditions and standardized request patterns.

Each experimental scenario was executed through repeated independent benchmark runs. Cold, warm, and full-stack load scenarios were executed 20 times, whereas infrastructure-only load (mock) scenarios were executed 30 times due to their lower execution cost. Independent executions were separated by enforced idle periods in cold-start experiments, while warm and load scenarios followed predefined workload configurations. Reported latency values correspond to the median obtained from repeated measurements, while p95 latency was used to characterize tail behavior.

IV. Results and Analysis

The results of the experimental evaluation are analyzed through a comparative assessment of full-stack and infrastructure-only execution scenarios across different workload conditions. In addition to absolute latency values, particular emphasis is placed on the relative contribution of individual latency components, in accordance with the proposed decomposition model (Fig. 1) and its formal definition in (1).

Fig. 2 summarizes the relative contribution of latency components across the evaluated scenarios.

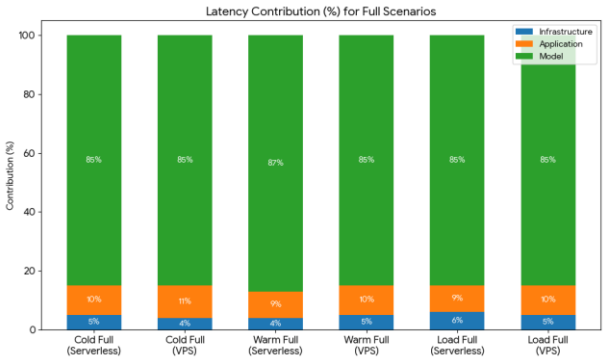

Fig. 2. Relative contribution of latency components (infrastructure, application, and model) across full-stack scenarios (Cold Full, Warm Full, Load Full) for Serverless and VPS platforms

The results show a clear dominance of the model layer, which consistently accounts for more than 85% of total response time in all scenarios. In contrast, infrastructure and application layers contribute only marginally, typically ranging between 4% and 11%.

Furthermore, the comparison between Serverless and VPS platforms reveals that their relative latency distributions are nearly identical in percentage terms. This indicates that, despite differences in deployment environments, the overall performance profile is primarily governed by the model component rather than infrastructure-related factors.

A. Full-Stack Performance Analysis

In the full-stack scenario, which includes the complete processing pipeline with external model invocation, both deployment environments exhibit consistently high response times across all workload conditions. Median latency values remain in the order of several seconds, with increased variability observed in tail latency metrics under load.

Despite differences in deployment paradigms, the overall latency profiles of serverless and VPS environments are relatively similar. While the VPS implementation achieves slightly lower median and tail latency values, the magnitude of this difference is limited. This observation suggests that infrastructure-related factors do not dominate system performance in full-stack AI scenarios.

A more detailed analysis of latency decomposition reveals that the model layer accounts for the majority of total response time, contributing approximately 85% of overall latency across all full-stack scenarios. In contrast, infrastructure and application-level processing together contribute less than 15%. This distribution directly explains the limited impact of infrastructure differences on total system performance.

These findings provide strong empirical support for the first hypothesis, confirming that in AI-driven full-stack scenarios, model-related latency dominates the execution path and effectively masks infrastructure-level variability.

B. Infrastructure-Only Performance Analysis

In the infrastructure-only scenario, where the model layer is replaced with a deterministic response, the latency profile changes significantly. Overall response times are drastically reduced, allowing infrastructure-related effects to become clearly observable.

In the infrastructure-only scenario, infrastructure and application layers together account for 100% of total latency, with infrastructure contributions ranging between 55% and 75%, depending on workload conditions and deployment environment. In this context, the VPS implementation consistently outperforms the serverless deployment, achieving lower latency values and more stable performance across all scenarios.

The difference is particularly pronounced in cold-start conditions, where serverless execution exhibits significantly higher latency due to initialization overhead. Under load conditions, serverless deployments also demonstrate increased variability, indicating the presence of scaling and orchestration overheads. In contrast, the VPS environment maintains more consistent latency characteristics, reflecting its persistent execution model.

These results confirm the second hypothesis, demonstrating that infrastructure becomes a dominant performance factor when model-related latency is removed from the system.

C. Comparative Latency Decomposition

The combined analysis of full-stack and infrastructure-only scenarios enables a comprehensive interpretation of latency contributions within the proposed decomposition framework. As illustrated in Fig. 1 and supported by the experimental results, system latency exhibits a dual behavior depending on the presence of external model processing.

$$\frac{T_{(model)}}{T_{(total)}} >> \frac{T_{infra}+T_{app}}{T_{(total)}} \quad (2)$$

As expressed in (2), the relative contribution of the model layer significantly exceeds the combined contribution of infrastructure and application layers in full-stack scenarios. This inequality formalizes the empirical observation that model-related latency dominates the execution path in AI-driven applications.

In full-stack scenarios, the model component overwhelmingly dominates total latency, rendering infrastructure-level differences largely insignificant from an end-user perspective. Conversely, in infrastructure-only scenarios, the absence of model latency exposes the underlying impact of infrastructure and application processing, making performance differences between deployment environments clearly measurable.

This duality highlights a critical limitation of traditional benchmarking approaches. Evaluations that focus exclusively on infrastructure may produce misleading conclusions when applied to AI-driven systems, as they fail to account for the dominant influence of external model services.

D. Implications for System Design

The results suggest that performance evaluation in modern AI-powered web applications must move beyond infrastructure-centric benchmarking toward a system-level perspective. The quantitative evidence presented in the experimental results demonstrates that optimizing infrastructure alone is unlikely to yield substantial improvements in overall latency when model inference dominates the execution pipeline.

Instead, performance optimization should focus on reducing model-related latency and improving orchestration efficiency. This includes strategies such as response streaming, caching of model outputs, request batching, and minimizing unnecessary interactions with external services.

Furthermore, the findings indicate that infrastructure selection should be context-dependent. While serverless architectures may remain suitable for applications with variable workloads, and VPS deployments may offer advantages in latency stability, their relative importance diminishes in scenarios where external AI services govern the critical path.

E. Tail Latency and Variability Analysis

The analysis of tail latency further reinforces the observed trends. As shown in Fig. 2, the difference between deployment environments becomes significantly more pronounced when considering p95 latency values. In full-stack scenarios, both platforms exhibit high variability due to model-related delays, with only minor differences between them. However, in infrastructure-only scenarios, serverless deployments demonstrate substantially higher tail latency compared to VPS, indicating less stable performance under load.

This observation suggests that infrastructure-related overheads primarily affect latency variability rather than average response time in AI-driven systems. Consequently, tail latency metrics provide a more sensitive indicator of infrastructure performance when model latency is not dominant.

V. DISCUSSION

The results of this study provide important insights into the evolving nature of performance bottlenecks in modern AI-driven web applications. While traditional research has emphasized infrastructure as the primary determinant of system performance, the findings presented here suggest a shift toward model-centric latency dominance in applications that rely on external AI services.

A key observation emerging from the analysis is that, in full-stack scenarios, the contribution of infrastructure to total latency becomes marginal when compared to the latency introduced by external model inference. This finding aligns with recent research on AI application pipelines, which highlights that performance in systems based on retrieval-augmented generation and large language models is shaped by multi-stage processing workflows rather than isolated system components [10], [18]. In such architectures, latency introduced by model inference and external API communication represents a major contributor to overall response time and is known to introduce significant delays in LLM-based systems [19].

From the perspective of serverless computing, these findings provide a nuanced interpretation of previously reported performance characteristics. Prior studies have shown that serverless platforms exhibit variability due to cold-start latency, resource allocation policies, and orchestration overhead [7], [8]. However, the results of this study indicate that the practical impact of these factors depends strongly on the application context. When model latency dominates, infrastructure-level variability becomes less relevant to the overall system behavior. This suggests that conclusions drawn from infrastructure-only benchmarks may not generalize to AI-driven systems.

Furthermore, the observed differences in infrastructure-only scenarios are consistent with existing literature on serverless performance limitations. The higher latency and variability observed in serverless environments under cold and load conditions reflect known challenges related to function initialization and distributed execution overhead [8], [9]. In contrast, the more stable performance of VPS-based systems

corresponds to the "always-on" execution model typically associated with traditional server-based architectures. These results confirm that infrastructure remains a critical factor in scenarios where model-related latency is absent or negligible.

Taken together, these findings highlight the importance of adopting a system-level perspective when analyzing performance in AI-enabled web applications. Rather than treating infrastructure and model components as independent concerns, performance must be understood as the result of their interaction within a distributed processing pipeline. This perspective is supported by recent work on serverless data systems and vector databases, which emphasizes the role of data access patterns, communication overhead, and orchestration in shaping overall system latency [11].

The implications of this study extend beyond performance evaluation to system design and optimization strategies. Traditional approaches that prioritize infrastructure optimization such as selecting between serverless and VPS deployments may yield limited benefits in AI-driven applications if the dominant source of latency lies in external model interaction. Instead, optimization efforts should focus on reducing model-related latency through techniques such as caching, request batching, and improved orchestration of AI workflows.

At the same time, it is important to acknowledge the limitations of the presented study. The experimental setup is based on a specific application architecture and a particular class of AI workloads, which may influence the generalizability of the results. Additionally, factors such as network conditions, model provider performance, and workload variability can further affect latency characteristics in real-world systems.

Overall, this study contributes to a growing body of evidence that challenges infrastructure-centric views of performance in modern web systems. By demonstrating that latency in AI-driven applications is often dominated by model-related components, the results underscore the need for a paradigm shift toward end-to-end performance analysis and system-level optimization.

VI. Limitations and Future Work

While the results of this study provide valuable insights into the performance characteristics of AI-driven web applications, several limitations should be acknowledged.

First, the experimental evaluation is based on a specific application architecture and a single class of AI workload relying on external large language model services. Although the selected setup reflects a realistic and representative use case, different application patterns such as streaming-based interactions, multi-agent systems, or locally hosted models may exhibit different latency characteristics. As a result, the generalizability of the findings may be influenced by the nature of the application and the structure of the processing pipeline.

Second, the study considers a single external model provider, whose performance characteristics including response time variability and service-side optimization directly affect the observed results. Since external AI services represent a major component of total latency, differences between model providers or deployment configurations (e.g., regional endpoints or model sizes) could lead to variations in the relative contribution of latency components.

Third, the experimental setup assumes controlled network conditions and a fixed testing environment. In real-world scenarios, network variability, geographical distribution of users, and dynamic workload patterns may introduce additional sources of latency that are not fully captured in this study. In particular, cross-region communication and fluctuating network performance could significantly influence end-to-end response times.

Fourth, while the proposed latency decomposition framework separates infrastructure, application, and model-related components, it does not explicitly account for all internal sub-processes within each layer. For example, retrieval operations, caching mechanisms, and data preprocessing steps are treated as part of the application layer, although they may introduce distinct performance behaviors that warrant further analysis.

Despite these limitations, the study establishes a foundation for a more comprehensive understanding of performance in AI-powered web systems. Future work should extend the proposed framework to a broader range of architectures and deployment scenarios. In particular, evaluating systems that incorporate edge computing, hybrid cloud environments, or on-device inference could provide deeper insights into the interplay between infrastructure and model-related latency.

Additionally, future research should explore adaptive optimization strategies that dynamically balance system components based on workload characteristics. This includes techniques such as intelligent request routing, caching of model responses, partial inference, and asynchronous processing. Integrating these strategies into the latency decomposition framework could enable more efficient system design and real-time performance optimization.

Finally, further investigation is needed to quantify the impact of emerging AI system paradigms, such as agent-based architectures and multi-step reasoning pipelines, on end-to-end latency. As AI applications continue to evolve, performance analysis must also adapt to account for increasingly complex interactions between distributed system components.

VII. Conclusion

This paper presented an empirical analysis of performance in AI-driven web applications through the lens of end-to-end latency decomposition. Unlike traditional infrastructure-centric evaluations, the proposed approach explicitly separates infrastructure-related, application-level, and model-related contributions to total response time, enabling a more precise understanding of system behavior.

The experimental results demonstrate that, in full-stack AI scenarios, the dominant portion of latency originates from the

model layer, while infrastructure-related differences have a comparatively limited impact on overall performance. Conversely, in infrastructure-only scenarios, the influence of deployment environments becomes clearly observable, confirming that infrastructure remains a relevant factor when model-related latency is absent. These findings highlight the conditional nature of infrastructure impact and challenge the assumption that infrastructure optimization alone is sufficient for improving performance in modern AI systems.

The main contribution of this work lies in reframing performance analysis from an infrastructure-centric to a system-level perspective. By introducing a latency decomposition framework and validating it through controlled experiments, the study provides a practical methodology for identifying performance bottlenecks in AI-powered applications.

From a practical standpoint, the results suggest that optimization strategies should prioritize model interaction and orchestration efficiency rather than focusing exclusively on backend infrastructure. Techniques such as caching, request optimization, and improved coordination of distributed components are likely to yield more significant performance gains in AI-driven environments.

Future work should extend this analysis to a broader range of application types, deployment platforms, and AI models, as well as explore hybrid and edge-based architectures. Further investigation into adaptive optimization strategies that dynamically balance infrastructure and model-level performance may provide additional insights into the design of next-generation AI systems.

Beyond the immediate findings, this work also contributes to a broader understanding of performance evaluation in the context of increasingly complex AI ecosystems. As modern applications evolve toward distributed, model-centric architectures, traditional evaluation methodologies that isolate infrastructure performance become insufficient. The results of this study highlight the need for integrated performance models that capture interactions between infrastructure, application logic, and external AI services.

In this regard, the proposed latency decomposition framework can serve as a foundation for future research on performance-aware system design. It enables not only the identification of dominant latency sources, but also supports the development of adaptive optimization strategies tailored to specific workload characteristics. Such strategies may include dynamic routing of requests, hybrid deployment models, and intelligent orchestration mechanisms that balance infrastructure efficiency with model responsiveness.

Ultimately, as AI systems continue to integrate more tightly with real-time and large-scale applications, understanding and managing end-to-end latency will become a critical design requirement rather than an optimization afterthought. The insights presented in this paper provide a step toward that goal, offering both a conceptual and empirical basis for rethinking performance in AI-driven web systems.

REFERENCES

[1] S. Gupta, R. Ranjan, and S. N. Singh, "A comprehensive survey of retrieval-augmented generation (RAG): Evolution, current landscape and future directions," *arXiv preprint arXiv:2410.12837*, 2024.

[2] A. A. Khan, M. T. Hasan, K. K. Kemell, J. Rasku, and P. Abrahamsson, "Developing retrieval augmented generation (RAG) based LLM systems from PDFs: An experience report", *arXiv preprint arXiv:2410.15944,* 2024.

[3] M. Zaharia et al., "Accelerating the Machine Learning Lifecycle with MLflow," *IEEE Data Engineering Bulletin*, 41(4), pp. 39–45, 2018.

[4] I. Srivastava, and Deepshikha. "A review of comparative study of serverless computing and virtual machines in cloud environments", *International Research Journal of Engineering and Technology (IRJET)*, 12(4), pp. 692–697. 2025, https://www.irjet.net/archives/V12/i4/IRJET-V12I4108.pdf?utm_source

[5] V. Fujiyanti, G. M. Suranegara, and I. N. Ichsan, "Comparative analysis of server-based and serverless service performance on Google Cloud Platform (GCP): Case study of machine learning model deployment", *Journal of Information Systems and Informatics*, 6(2), pp. 1172–1194, 2024.

[6] S. Henning and W. Hasselbring, "A configurable method for benchmarking scalability of cloud-native applications," *Empirical Software Engineering,* 27(6), article 143, 2022.

[7] J. Scheuner, S. Eismann, S. Talluri, E. van Eyk, C. Abad, P. Leitner, and A. Iosup, "Let's trace it: Fine-grained serverless benchmarking using synchronous and asynchronous orchestrated applications", *arXiv preprint arXiv:2205.07696,* 2022.

[8] J. Scheuner et al., "TriggerBench: A performance benchmark for serverless function triggers," *in Proc. IEEE Int. Conf. on Cloud Engineering (IC2E)*, pp. 96–103, 2022.

[9] 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 Computing Surveys,* 2025.

[10] E. Jonas et al., "Cloud programming simplified: A Berkeley view on serverless computing," *arXiv preprint arXiv:1902.03383*, 2019.

[11] P. Castro, V. Ishakian, V. Muthusamy, and A. Slominski, "The rise of serverless computing," *Commun. ACM*, 62(12), pp. 44–54, 2019.

[12] J. Scheuner, "Benchmarking and performance analysis of serverless computing platforms," *PhD dissertation, Chalmers Univ.*, 2022.

[13] A. van Eyk et al., "The SPEC-RG reference architecture for FaaS: From microbenchmarks to applications," *IEEE Cloud Computing*,. 6(3), pp. 7–18, 2019.

[14] G. Wang et al., "Peeking behind the curtains of serverless platforms," in *Proc. USENIX ATC*, pp. 133–146, 2018.

[15] M. Shahrad et al., "Serverless in the wild: Characterizing and optimizing the serverless workload at a large cloud provider," in *Proc. USENIX ATC*, 2020.

[16] I. Akkus et al., "SAND: Towards high-performance serverless computing," in *Proc. USENIX ATC*, pp. 923–935, 2018.

[17] Y. Su, Y. Sun, M. Zhang, and J. Wang, "Vexless: A serverless vector data management system using cloud functions," *Proc. ACM on Management of Data,* 2(3), article 187, 2024.

[18] I. Baldini et al., "Serverless computing: Current trends and open problems," in *Research Advances in Cloud Computing, Springer*, pp. 1-20, 2017.

[19] X. Miao et al., "Towards Efficient Generative Large Language Model Serving: A Survey from Algorithms to Systems," *arXiv preprint arXiv:2312.15234, 2023*

Jasna Hamzabegović is an Associate Professor at the Faculty of Technical Sciences, University of Bihać, Bosnia and Herzegovina. She received her B.Sc. degree in Informatics from the University of Sarajevo, the M.Sc. degree in Computer Science and Informatics from the University of East Sarajevo, and the Ph.D. degree in Technical Sciences from the University of Bihać in 2014. Her research interests include educational software development, digital literacy, game-based learning, and user-centered applications for vulnerable populations.

Dr. Hamzabegović has authored or co-authored approximately 40 scientific and professional publications and is the co-author of the university textbook Object-Oriented Programming with C++. She has participated in several international projects in the areas of innovative education and digital transformation and serves as a reviewer for international conferences. She is currently the Head of the Department of Electrical Engineering at the Faculty of Technical Sciences, University of Bihać

Amel Džanić is a Senior Teaching Assistant at the Faculty of Technical Engineering, University of Bihać. He graduated in February 2003 from the Faculty of Electrical Engineering, University of Sarajevo. He completed his Master's degree at the Faculty of Technical Engineering, University of Bihać, in the Department of Electrical Engineering, majoring in Computer Science. He has published over 20 scientific and professional papers as an author or co-author, as well as university textbooks Object-Oriented Programming wirh C++ and Design of Experiment. He has participated in several projects in the fields of computer science and information technology.
His areas of scientific and professional interest include software engineering and information systems, as well as programming and programming languages.

Kenan Duraković received his B.Sc. degree in Electrical Engineering, majoring in Computer Science and Informatics, from the University of Bihać, Bosnia and Herzegovina, in 2025. He graduated as the top student of his generation, demonstrating outstanding academic performance throughout his undergraduate studies.

His research interests include artificial intelligence, software engineering, web application development, distributed systems, performance analysis, and information systems. He is the author of a scientific paper focusing on the performance evaluation of modern AI-powered software systems.

During his studies and professional engagement as a freelance software developer, he designed and implemented several software solutions for real-world business applications, including enterprise information systems, production management platforms, inventory management systems, invoicing solutions, and AI-powered web applications. His work combines modern web technologies with practical business requirements, emphasizing scalability, automation, and user-centered software design