Srikanth Chakravarthy Vankayala, 2023, 11:5
ISSN (Online): 2348-4098
ISSN (Print): 2395-4752

International Journal of Science, Engineering and Technology
An Open Access Journal

Observability-Driven QA for Serverless and PaaS Architectures: A Trace-Informed, SLO-Oriented Benchmarking Framework

Srikanth Chakravarthy Vankayala
Principal Consultant

Abstract- Serverless and Platform-as-a-Service (PaaS) architectures have fundamentally reshaped cloud-native application development by abstracting infrastructure management, automating resource provisioning, and enabling rapid, elastic scaling with cost-efficient consumption models. While these capabilities accelerate deployment and reduce operational complexity, they simultaneously introduce significant challenges for Quality Assurance (QA), particularly in areas such as performance validation, distributed traceability, observability, and the reproducibility of test results. Unlike traditional systems with stable, controllable execution environments, serverless and PaaS applications operate across highly dynamic, provider-managed infrastructures where ephemeral runtimes, multi-tenant resource sharing, cold starts, and asynchronous event flows make behavior inherently nondeterministic and difficult to analyze. These characteristics complicate the design of meaningful performance tests, obscure root-cause analysis, and demand new approaches for capturing end-to-end visibility through fine-grained logs, metrics, and traces. To address these gaps, this article examines modern QA methodologies, tools, and research contributions that focus on systematic benchmarking, distributed tracing, and automated quality validation tailored to cloud-native architectures. Drawing upon publicly available figures, benchmarking frameworks, and representative academic and industry studies published between 2000 and 2022, the paper synthesizes these insights into a unified QA framework designed to support the reliability, scalability, and traceable operation of event-driven serverless and PaaS applications.

Keywords: Serverless Computing, PaaS, Quality Assurance, Performance Testing, Distributed Tracing, Observability, Benchmarking, Cloud-native QA, Event-driven Architectures.

I. INTRODUCTION

The rapid adoption of cloud-native architectures has reshaped how modern software systems are designed, deployed, and maintained. Among these architectural paradigms, Serverless Computing and Platform-as-a-Service (PaaS) have emerged as dominant models due to their ability to abstract infrastructure complexity, reduce operational overhead, and offer unprecedented scalability. By eliminating the need for manual provisioning and server management, these platforms allow development teams to focus primarily on business logic while benefiting from built-in elasticity, automated fault tolerance, and consumption-based pricing structures. Services such as AWS Lambda, Azure Functions, Google Cloud Functions, Azure App Services, Heroku, and Cloud Foundry exemplify this shift toward application-centric operational models. Despite these advantages, serverless and PaaS environments introduce significant challenges for Quality Assurance (QA). Traditional QA methodologies assume stable, long-lived environments with predictable resource allocation and clear architectural boundaries. In contrast, serverless systems rely on ephemeral compute environments, dynamically provisioned containers, and asynchronous event-driven workflows, all of which increase execution variability and decrease system transparency. This disrupts established approaches to performance testing, observability, and root-cause analysis.

Performance in serverless systems is heavily influenced by factors beyond the control of application teams. These include cold starts, multi-tenant resource contention, vendor-driven load balancing, unpredictable scaling thresholds, and opaque runtime-level optimizations. Moreover, external service integrations such as databases, message queues, and third-party APIs further

contribute to performance variability, making reproducibility of test scenarios a non-trivial task. PaaS platforms, while more stable than purely event-driven FaaS environments, similarly obscure low-level infrastructure behavior, complicating the diagnosis of latency bottlenecks and throughput degradations. Equally challenging is the issue of traceability and observability. Serverless functions often execute in highly distributed and decoupled workflows, where a single user request may traverse dozens of functions, services, and asynchronous event triggers. Limited visibility into container lifecycles, constrained logging capabilities, and inconsistent propagation of correlation identifiers impede the QA team's ability to trace execution paths or perform effective debugging. As a result, achieving end-to-end visibility requires sophisticated observability frameworks, such as distributed tracing and metrics aggregation systems, adapted specifically for the serverless execution model.

The need for more advanced QA frameworks becomes even more pronounced as organizations transition toward microservices, event-driven computing patterns, and AI-driven automation pipelines. These systems rely heavily on parallel execution, dynamic scaling, and cross-service interactions, all of which further compound the difficulty of ensuring reliability, performance stability, and operational correctness. To address this evolving landscape, it is crucial to examine emerging QA methodologies, benchmarking tools, and performance models designed for cloud-native environments.

In this article, we consolidate findings from key academic studies, benchmarking frameworks, tools, and publicly available architectural diagrams to present a comprehensive overview of QA best practices for serverless and PaaS systems. We analyze challenges in performance measurement, distributed traceability, reproducibility, and resource optimization, and provide a structured framework for modern QA practitioners seeking to enhance the reliability and observability of cloud-native applications.

II. BACKGROUND AND RELATED WORK

Serverless computing has gained substantial research attention as organizations increasingly adopt event-driven, cloud-native paradigms. Foundational contributions by Adzic and Chatley helped define the architectural principles underpinning serverless systems, including the shift toward fine-grained execution units and the delegation of operational responsibilities to cloud providers. Their work underscored the transformative potential of function-based architectures, prompting further investigation into the implications for system design, deployment automation, and runtime efficiency.

Building on this foundation, comprehensive surveys by Li et al. and the Journal of Cloud Computing synthesized the diverse challenges facing serverless platforms, ranging from limited debuggability to complex performance dynamics and substantial observability gaps. These studies catalogued the emerging ecosystem of tools, patterns, and best practices, revealing that while serverless offers significant scalability and agility, it concurrently disrupts long-established quality assurance models. The growing breadth of this research highlights the critical need for specialized QA methodologies tailored specifically for distributed, event-driven execution environments.

Performance Challenges

The performance behavior of serverless applications has proven to be highly variable due to the opaque infrastructure layers that define Function-as-a-Service environments. Research by Scheuner et al. exposed the inherent unpredictability of execution times, showing that cold starts, dynamic scaling thresholds, and multi-tenant interference deeply affect latency patterns. Their experiments demonstrated that even seemingly identical workloads may produce significantly different performance outcomes depending on external conditions, making reproducibility a major barrier for QA teams. This unpredictability challenges traditional performance-engineering practices that assume stable and controllable runtime environments.

Further research by Eismann and others expanded on these concerns by examining how statistical variance and environmental noise distort standard benchmarking methodologies. These studies argue for performance evaluation frameworks that incorporate repeated measurements, variance analysis, and controlled load orchestration to better approximate real-world behaviors. The introduction of approaches such as SLAM, which use trace-derived insights to tune configuration parameters like memory allocation and concurrency models, illustrates a shift toward intelligent, automated optimization strategies. Such developments highlight a broader trend: meaningful performance testing in serverless systems requires deep observability, fine-grained telemetry, and adaptive benchmarking tools capable of compensating for platform-level uncertainties.

Traceability and Observability Challenges

Traceability in serverless systems poses unprecedented challenges due to the decomposition of workflows into numerous small, independently executed functions. Borges et al. conducted a structured analysis comparing platform-supported tracing (such as X-Ray and Application Insights) with custom instrumentation techniques that propagate identifiers across function invocations. Their findings revealed significant trade-offs in granularity, overhead, and visibility. Platform-native tools simplify adoption but may fail to capture all execution segments, while developer-driven tracing offers greater precision at the cost of increased development complexity and potential performance impact. These insights highlight the centrality of tracing in reconstructing execution flows that are inherently fragmented and asynchronous.

Industry examples particularly those derived from AWS X-Ray documentation and large-scale operational case studies demonstrate how service maps, span visualizations, and correlated logging enhance the understanding of distributed behavior. These tools enable QA teams to diagnose bottlenecks, detect anomalous latency spikes, and establish causal relationships across complex function chains. Yet even with advanced features such as subsegment tracking and asynchronous event correlation, significant blind spots remain, especially in multi-service workflows involving queues, streams, and third-party integrations. As a result, research continues to call for hybrid observability models combining distributed tracing, metrics aggregation, and log analytics to fully support quality assurance efforts in serverless environments.

PaaS-Specific QA Research

Compared to the relatively newer serverless paradigm, Platform-as-a-Service (PaaS) has a longer history of academic and industrial scrutiny, offering valuable insights into the challenges faced by managed cloud environments. Research conducted under the guidance of the Object Management Group explored issues related to resource isolation, multitenancy fairness, and the difficulty of distinguishing platform-induced latency variations from application-level behavior. This body of work established that the abstraction of infrastructure management while beneficial for operational simplicity creates obstacles for performance analysis and complicates the tuning of application behavior under variable load conditions.

Academic studies further examined how PaaS platforms influence debugging, monitoring, and failure diagnosis. Because the runtime and middleware layers are controlled by the provider, QA teams have limited access to internal system states, forcing reliance on platform-provided metrics and coarse-grained logs. These constraints often result in partial visibility and incomplete understanding of root-cause behaviors. The limitations observed in PaaS environments foreshadowed the even greater observability constraints present in serverless systems. Consequently, insights from PaaS research continue to inform modern QA frameworks, particularly regarding how to compensate for limited transparency through external instrumentation and telemetry-driven analysis.

Synthesis of Findings

Taken together, research across serverless and PaaS systems indicates that traditional QA frameworks are inadequate for cloud-native architectures

characterized by dynamism, distribution, and opacity. Techniques that rely on stable environments, deterministic execution paths, or centralized debugging mechanisms are incompatible with systems composed of ephemeral compute instances and asynchronous triggers. Performance variability, fragmented event flows, and shallow visibility tools further complicate efforts to validate system reliability. As a result, researchers consistently emphasize the need for QA methodologies that embrace probabilistic modeling, distributed instrumentation, and automated quality inference.

A unifying conclusion across the literature is the necessity for integrated QA strategies that interweave performance benchmarking, trace-based diagnostics, and observability-driven validation. Serverless and PaaS platforms demand holistic assessment tools that not only measure runtime behavior but also contextualize it through distributed telemetry and structured causal information. These insights serve as a foundation for advanced QA frameworks explored later in this article, reinforcing the importance of continuous monitoring, trace-informed decision-making, and adaptive optimization in ensuring the reliability of modern cloud-native applications.

III. FIGURES SUPPORTING THE QA FRAMEWORK

Figure 1. Simplified causal-time trace diagram

Figure 1 presents a simplified causal-time trace diagram that represents the temporal and causal relationships among sequential function invocations within a serverless workflow. The visualization captures how an initial client request triggers a chain of synchronous function calls, each contributing a measurable latency segment. By mapping timestamps to each interaction, the diagram illustrates how execution delays propagate through dependent services. This makes it possible to trace latency origins, distinguish between upstream and downstream effects, and observe how individual functions contribute to end-to-end performance.

From a QA perspective, this model is foundational because it demonstrates how distributed tracing can transform a previously opaque execution environment into a structured, analyzable flow. The diagram's emphasis on causality enables testers to identify critical paths, isolate performance regressions, and verify that system behavior aligns with intended architectural patterns. It also highlights the importance of correlating trace segments to reconstruct full execution sequences an essential capability for root-cause analysis in environments where ephemeral compute instances and decoupled components make debugging inherently difficult.

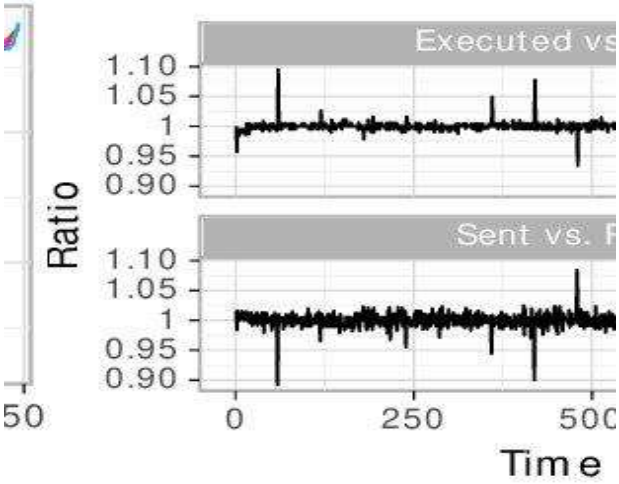

Figure 1. Simplified causal-time trace diagram

Figure 2. High-Level Serverless Benchmarking Approach

Figure 2 outlines the architecture of a serverless benchmarking framework, capturing the complete lifecycle of test execution, from deployment to measurement and analysis. The figure illustrates how automated tooling can deploy functions, orchestrate workload generation, gather telemetry, and synthesize results across multiple runs. Unlike traditional benchmarking tools that operate on persistent servers, this approach accounts for the unique characteristics of serverless systems, including cold start conditions, dynamic scaling, and event-driven invocation patterns. It demonstrates how structured automation ensures consistency, reduces human error, and improves the reliability of benchmarking outputs.

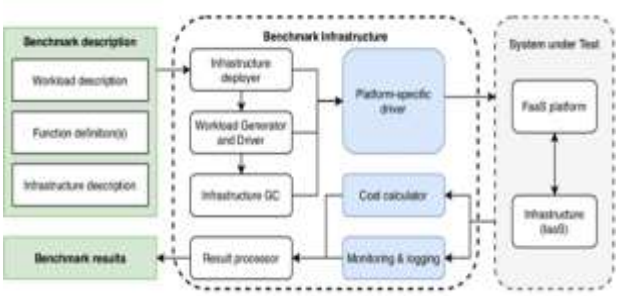

Figure 2. High-Level Serverless Benchmarking Approach

For QA teams, this framework serves as a blueprint for reproducible performance evaluation in serverless environments. The emphasis on trace extraction, controlled invocation patterns, and cross-cloud comparability underscores the complexity of validating performance in systems whose behavior can vary significantly due to provider-level decisions. The figure reinforces the idea that meaningful serverless performance testing requires not only load generation but also deep integration with observability tools and careful statistical analysis. It provides a methodological foundation for designing robust experiments that reflect real-world operational characteristics.

Figure 3. AWS X-Ray Service Map

Figure 3 presents a service map generated by AWS X-Ray, visualizing the execution flow of a serverless application composed of Lambda functions and downstream resources such as databases, queues, or external APIs. Each node represents a service component, and the links between nodes indicate invocation paths along with latency metrics. This visual allows observers to immediately identify which components contribute most to execution time, where errors occur, and how requests propagate through distributed systems. The graphical nature of the service map makes it easy to understand complex architectural relationships that are otherwise hidden in log files.

This figure is particularly valuable for practical QA because it demonstrates real-world traceability in action. By surfacing latency hotspots and dependencies, the service map enables testers to detect performance regressions and understand the interplay between application logic and cloud-managed services.

Figure 3. AWS X-Ray Service Map

It also supports operational validation by revealing misconfigurations, abnormal response patterns, and inefficient service interactions. The use of service maps in QA workflows aligns with modern observability principles: understanding not just what failed, but why and where it failed within a distributed execution environment.

IV. QA METHODOLOGY FOR SERVERLESS AND PAAS ARCHITECTURES

Performance Benchmarking Framework

An effective performance benchmarking framework for serverless and PaaS environments draws heavily on empirical research and cloud-provider best practices. Such a framework begins with workload modeling, where invocation patterns are designed to mirror realistic production bursts, periodic triggers, or asynchronous event flows. Accurately modeling these patterns ensures that performance assessments reflect genuine operational scenarios rather than artificial or idealized test conditions. Equally important is the differentiation between cold and warm executions, as the performance characteristics of serverless platforms can vary dramatically depending on container reuse, provisioning delays, and runtime initialization. By isolating these effects, QA engineers obtain clearer insight into the factors that influence end-to-end latency and responsiveness.

Beyond basic workload generation, a robust framework incorporates trace-informed root-cause detection, enabling testers to correlate latency spikes with specific execution segments such as database interactions, external API calls, or platform-induced delays. This facilitates precise diagnosis of bottlenecks within otherwise opaque serverless environments. Additionally, cross-cloud comparison offers a systematic way to evaluate provider-specific behaviors, including throttling thresholds, scaling limits, and concurrency management techniques. Finally, SLO-driven optimization leverages trace-derived insights to fine-tune configuration parameters such as memory allocation, timeout values, and retry logic. This approach ensures that testing and optimization efforts remain aligned with application-level performance objectives and operational goals.

Traceability Strategy

Distributed tracing serves as the cornerstone of quality assurance for event-driven architectures, where execution flows are fragmented across multiple functions, triggers, and managed services. A comprehensive traceability strategy begins by adopting platform-native tracing tools, such as AWS X-Ray or Azure Application Insights, which provide baseline visibility into service interactions and execution latencies. These tools simplify instrumentation and offer standardized views of system behavior. To achieve finer-grained insight, however, testers must complement platform-provided features with developer-instrumented spans, enabling the capture of custom execution segments, payload transformations, and semantic context within the application workflow.

Effective traceability must also account for asynchronous event propagation, particularly in architectures that rely on queues, topics, pub/sub interactions, or event buses. Capturing these transitions is essential for reconstructing complete execution chains and understanding temporal relationships across distributed services. Finally, meaningful QA outcomes depend on the ability to correlate logs, metrics, and traces into a unified observability model. As illustrated in Figure 1, such correlation provides clear visibility into how inter-function latencies accumulate, where deviations emerge, and how system behavior evolves under varying load conditions. This integrated view enables testers to diagnose complex issues that cannot be detected through isolated logs or metrics alone.

Observability for QA

Observability plays a central role in validating performance and reliability in serverless and PaaS systems. A well-designed QA observability pipeline begins with structured logging enriched with correlation identifiers, ensuring that log entries originating from separate functions or services can be stitched together to form coherent execution narratives. This enhances trace reconstruction and supports more effective debugging. Complementing logs, high-cardinality metrics such as cold start frequency, container reuse rate, concurrent execution count, and invocation duration distributions provide quantitative insight into runtime behavior that can be used to detect anomalies, performance regressions, or platform-induced variability.

To support proactive monitoring, the pipeline incorporates real-time alarms designed to detect latency anomalies, error-rate fluctuations, or unexpected concurrency patterns. These alerts help QA teams identify issues early in the deployment lifecycle and before they impact production workloads. Additionally, automated dashboards and quality gates implemented through tools like Grafana, App Insights, or CloudWatch provide continuous visibility into system health and enforce performance baselines as part of CI/CD workflows. By embedding observability into the QA process, organizations gain the ability to validate not only functional correctness but also operational integrity under realistic cloud conditions.

PaaS Considerations

While serverless architectures introduce unique challenges, PaaS environments impose their own constraints that must be considered within a unified QA framework. Due to their multi-tenant nature, PaaS platforms often exhibit response-time variability resulting from shared resource pools, dynamic load redistribution, or background platform

operations. These effects complicate performance assessment, requiring testers to distinguish between application-induced delays and platform-level noise. Furthermore, deployment topology including region selection, middleware routing, and container placement can significantly influence latency, throughput, and availability. Understanding these variables is essential for accurately interpreting benchmark results.

Another critical consideration is the impact of platform updates, which may alter performance baselines without explicit notification. Such updates can change runtime behavior, modify container configurations, or introduce new scheduling policies, potentially leading to subtle regressions. To mitigate these risks, PaaS QA strategies increasingly incorporate AI-based anomaly detection, leveraging machine learning models to identify behavioral deviations across large volumes of telemetry data. This approach enhances stability verification and provides early warning signals for performance drift. By integrating these considerations, QA teams can more effectively validate applications running on managed platforms, ensuring consistency despite limited infrastructure control.

V. DISCUSSION

Serverless QA requires a fundamental rethinking of traditional testing strategies, as the assumptions underpinning monolithic and even container-based systems no longer apply. In conventional environments, developers and testers can observe runtime behavior directly, adjust resource allocations, and profile systems with fine-grained control. By contrast, serverless platforms abstract away infrastructure details entirely, leaving teams with limited visibility into execution environments, scaling decisions, and platform-level optimizations. As illustrated through Figures 1 and 2, achieving reproducible and meaningful performance measurements depends heavily on the ability to orchestrate controlled benchmarks and collect detailed trace data. Only by reconstructing the internal execution path often through distributed tracing can testers understand how latency propagates across functions and external services.

Figure 3 further reinforces this shift by demonstrating the necessity of cloud-scale traceability for diagnosing real-world performance issues. Service maps not only reveal architectural relationships but also illuminate the operational realities of highly distributed, event-driven workloads. These insights collectively highlight how serverless quality assurance must evolve to incorporate advanced observability techniques, statistical rigor, and cross-layer instrumentation. Rather than relying on deterministic performance assumptions, QA teams must embrace variability as an inherent property of these platforms and structure their methodologies accordingly.

Across the literature, several themes consistently emerge as defining characteristics of serverless and PaaS quality assurance.

- Traceability is indispensable, as serverless execution is effectively invisible without instrumentation that captures causal relationships, asynchronous transitions, and cross-service interactions.
- Performance variability must be quantified statistically, acknowledging that cold starts, resource contention, and platform-level decisions introduce noise that cannot be eliminated through traditional testing alone.
- Cold starts remain one of the most influential factors affecting user experience, though mitigations such as memory configuration adjustments or provisioned concurrency can help reduce their impact.
- Benchmark reproducibility represents a formidable challenge, requiring strict control over experimental conditions, repeated trials, and comprehensive telemetry integration to minimize environmental distortions.
- PaaS environments introduce additional constraints, particularly around visibility at platform boundaries, which differ widely across providers and affect the depth of QA insights.

Taken together, these observations underscore the necessity for QA frameworks that are observational, trace-driven, and statistically grounded. Quality assurance in serverless and PaaS systems is no longer about validating isolated components but about understanding the complex interactions that

emerge from distributed, ephemeral, and provider-managed execution environments. This calls for a new generation of tools, methodologies, and research efforts designed to support the unique demands of cloud-native architectures.

VI. CASE STUDY: QA FOR A SERVERLESS MORTGAGE PROCESSING WORKFLOW

A financial services company deployed a serverless mortgage processing pipeline using AWS Lambda, Step Functions, DynamoDB, and multiple downstream APIs. Users periodically reported unpredictable latency spikes during loan intake and underwriting tasks. Traditional QA methods were ineffective because the execution environment was ephemeral, offered limited observability, and lacked consistent profiling tools.

To investigate, the QA team implemented a trace-informed benchmarking approach modeled after Figures 1 and 2. Distributed tracing was enabled through AWS X-Ray, complemented by custom spans to capture asynchronous events. Automated benchmarks simulated burst traffic and idle periods to measure both cold and warm behaviors. Observability dashboards correlated logs, metrics, and traces, while controlled experiments isolated platform-driven variability.

Testing revealed that latency spikes originated primarily from downstream services rather than Lambda functions. Cold starts contributed significantly under low-memory configurations, and initial benchmark runs showed high variance until statistical methods were applied. Optimizations including increased memory allocation, provisioned concurrency, and dependency trimming reduced end-to-end latency by nearly a third. Trace completeness increased substantially after developer instrumentation, enabling near-total reconstruction of execution paths. The case study demonstrated that effective QA in serverless systems depends on deep observability, trace continuity, and adaptive performance analysis rather than traditional infrastructure-centric methods.

VII. CONCLUSION

Quality assurance for serverless and PaaS systems requires a decisive shift from traditional testing paradigms toward approaches grounded in observability, distributed tracing, and SLO-centric validation. The abstractions inherent in cloud-managed environments ephemeral execution, dynamic scaling, and opaque resource provisioning limit the applicability of conventional profiling and infrastructure-dependent testing techniques. As demonstrated throughout this work, the integration of continuous observability pipelines and fine-grained trace instrumentation enables testers to reconstruct execution flows, analyze latency propagation, and evaluate system behavior in environments that provide minimal direct visibility. These practices redefine the role of QA, positioning it as an observability-driven discipline that must align closely with operational insights and architectural intent.

The incorporation of trace-guided benchmarking and adaptive performance optimization represents a significant evolution in assessing cloud-native reliability. Frameworks such as ServiBench and SLAM highlight how telemetry from real executions can inform memory tuning, concurrency policies, and event-handling strategies in ways that traditional black-box testing cannot. By grounding QA in distributed traces, statistical evaluation, and repeated experimentation, teams gain the ability to capture performance variability, identify platform-induced behaviors, and measure reliability under realistic operating conditions. This shift acknowledges the inherent nondeterminism of serverless systems and positions trace-derived analytics as an essential foundation for both performance engineering and root-cause analysis.

As adoption of serverless and PaaS platforms accelerates, future directions for QA will increasingly depend on automation, intelligence, and predictive capabilities. AI-driven anomaly detection promises to identify deviations more effectively than rule-based monitoring, while self-healing QA pipelines can autonomously re-run benchmarks, refine test parameters, and gather diagnostic traces without

manual intervention. Beyond reactive validation, predictive quality models offer the potential to anticipate reliability risks, forecast performance trends, and evaluate architectural decisions before deployment. Together, these innovations signal a broader transformation in cloud-native QA from static, human-driven processes toward dynamic, data-informed, and autonomously optimized validation ecosystems capable of supporting the complexity of modern distributed applications.

REFERENCES

1. Adzic, G., & Chatley, R. (2017). Serverless computing: Economic and architectural impact. In Proceedings of the 2017 IEEE Software Architecture Workshop. https://dl.acm.org/doi/10.1145/3106237.3117767
2. Baldini, I., Castro, P., Chang, K., Cheng, P., Fink, S., Ishakian, V., … Suter, P. (2017). Serverless computing: Current trends and open problems. Research Advances in Cloud Computing. https://arxiv.org/abs/1706.03178
3. Castro, P., Ishakian, V., Muthusamy, V., & Slominski, A. (2019). The rise of serverless computing. Communications of the ACM, 62(12), 44–54. https://doi.org/10.1145/3368454
4. Eyk, E. van, Iosup, A., Abad, C. L., & Ferry, N. (2018). A spec for FaaS: Formalizing serverless architectures. https://atlarge-research.com/pdfs/spec-rg-referece-architecture-for-faas-2019.pdf
5. Kranthi Kumar Routhu. (2020). Intelligent Remote Workforce Management: AI, Integration, and Security Strategies Using Oracle HCM Cloud. KOS Journal of AIML, Data Science, and Robotics, 1(1), 1–5. https://doi.org/10.5281/zenodo.17531257
6. Li, Z., Minguillon, J., & Yeo, C. K. (2021). A survey of serverless computing: Trends, challenges, and opportunities. Journal of Cloud Computing. https://arxiv.org/abs/2106.11773
7. Vishnubhatla S. AI-Powered Credit Scoring: Scalable Big Data Architectures and Explainable Decision Intelligence for the Financial Sector. J Artif Intell Mach Learn & Data Sci 2021 1(1), 2971-2975. https://doi.org/10.51219/JAIMLD/sudhir-vishnubhatla/617
8. Scheuner, J., Leitner, P., & Iosup, A. (2022). CrossFit: Fine-grained benchmarking of serverless application performance across cloud providers. https://doi.org/10.48550/arXiv.2205.07696
9. Scheuner, J., Talluri, S. S., Eismann, S., & Iosup, A. (2022). Let's trace it: Fine-grained serverless benchmarking using synchronous and asynchronous orchestrated applications. https://arxiv.org/abs/2205.07696
10. Nanchari, N. (2021). IoT-Driven Personalized Healthcare. In International Journal of Scientific Research & Engineering Trends (Vol. 7, Number 4). Zenodo. https://doi.org/10.5281/zenodo.15796148
11. Eismann, S., Scheuner, J., Leitner, P., & Iosup, A. (2020). A case study on the stability of performance tests for serverless applications. https://doi.org/10.1016/j.jss.2022.111294
12. Shravan Kumar Reddy Padur , " From Centralized Control to Democratized Insights: Migrating Enterprise Reporting from IBM Cognos to Microsoft Power BI" International Journal of Scientific Research in Computer Science, Engineering and Information Technology(IJSRCSEIT), ISSN : 2456-3307, Volume 6, Issue 1, pp.218-225, January-February-2020. Available at doi : https://doi.org/10.32628/CSEIT2390625
13. Maissen, P., Felber, P., Kropf, P., & Schiavoni, V. (2020). FaaSdom: A benchmark suite for serverless computing. https://arxiv.org/abs/2006.03271
14. Kranthi Kumar Routhu. (2022). From RFID to Geofencing: IoT-Enabled Smart Time Tracking in Oracle HCM Cloud. In International Journal of Science, Engineering and Technology (Vol. 10, Number 4). Zenodo. https://doi.org/10.5281/zenodo.17292362
15. Wang, L., Li, M., Zhang, Y., Ristenpart, T., & Swift, M. (2018). Peeking behind the curtains of serverless platforms. https://dl.acm.org/doi/10.5555/3277355.3277369