Literature Review for XAI-Driven Optimizing Cold Start Latency in Serverless Applications

N. Yaswanth, R. Rohan Chandra
Department of Computer Science and Engineering
National Institute of Technology Karnataka
Surathkal, Mangalore, India
Phone: +91-9676794131, +91-8639884378
namburiyaswanth.221cs232@nitk.edu.in, regulagaddarohanchandra.221cs241@nitk.edu.in

Abstract

Cloud computing has transformed the availability of information and communication technology (ICT) resources, transitioning from conventional infrastructure to a utility-oriented framework. While offering significant advantages, proficient administration of cloud resources necessitates a strong focus on requirements engineering, an area that is still relatively unexplored in scholarly works. This section presents an extensive review of the literature on requirements engineering in cloud computing, highlighting the necessity for a structured approach to address the multifaceted nature of cloud services. It identifies five critical dimensions—Contractual, Compliance, Financial, Operational, and Technical—that are essential for evaluating and specifying cloud requirements and supporting service level agreements (SLAs). The assessment found no clear norms or criteria for these measurements. Cold start delay is a significant issue in serverless computing, negatively impacting system performance. Recent research introduces a model that is integrated on an adaptive basis in the Software Development Lifecycle (SDLC) framework and seeks to enhance performance with deep learning and Long Short-Term Memory (LSTM) algorithms. It is established that such a model, tested on the major AWS Lambda and Microsoft Azure platforms, significantly helps in relieving the cold starts delay.MS. It reiterates the importance of predictive analytics in improving the performance of a serverless application and in sustaining the scale-to-zero characteristic.

I. Introduction

Cloud computing can be a paradigm shift because it enables businesses to effectively harness ICT related services and tools cooperatively. In the center of the internet, available on demand, cloud providers give the opportunity to many companies, to consider ICT facilities as any chemical like water, gas or electricity. This significantly affects how business and information technology are practiced. It eliminates the need for capital investments into physical hardware. However, the technological vision of cloud computing adoption is not well supported, this is due to how cloud computing requirements engineering is relatively undeveloped.

There are significant disparities between prior research on cloud computing and current research on cloud needs management that must be addressed in order to fill the gap. As far as standard software engineering used to emphasize time to market and feature sets, today's higher concentration on security and privacy aspects calls for changes as well. It is often necessary to create architectural elements that are sensitive to these. security and privacy concerns at the very initial stages to prevent reactive problem solving that is expensive and which takes time to implement.

The need for scalability and cost-effectiveness increased, there was a move from client/server architectures to cloud computing, which accelerated the development of varied service models such as Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS). Out of these, serverless computing is one of the options that have automatic scaling features however, cold-start delays are also one of the challenges. Performance issues like these have to be resolved in order to enhance optimization of serverless applications [1].

Recent studies suggest an adaptive model integrated in the Software Development Lifecycle (SDLC) that seeks to reduce cold-start latency through the use of predictive models. This model, focusing on LSTM and DNN algorithms, has demonstrated effective reduction. cold-start delay on major platforms - AWS Lambda and Microsoft Azure. This method provides better resource usage and still maintains the scale-to-zero feature of serverless computing by considering predictive models in the Software Development Lifecycle.

In cloud computing, managing both dynamic and static requirements effectively requires a comprehensive framework. The proposed dimensions—Contractual, Compliance, Financial, Operational, and Technical—provide a structured approach to link these aspects, ensuring that the needs of cloud consumers and providers are accurately aligned.Despite advances in security and privacy requirement engineering, there is still a shortage of integrated approaches that are customized to the specific characteristics of the cloud domain

This paper examines various requirements engineering methods applicable to both traditional and cloud architectures, highlighting the necessity for a unified framework that handles security and privacy issues in cloud environments.

To address these difficulties, this presentation will evaluate existing requirements engineering methods, focusing on both traditional and cloud-specific approaches. It initially assesses established methods that have typically addressed security and privacy separately, or considered privacy just as a subset of security. These approaches frequently fail to identify the interconnected nature of security and privacy requirements that is crucial in modern cloud environments.

The paper then explores methods specifically designed cloud computing emphasizes the necessity for a unified framework that addresses both security and privacy comprehensively.A structure like this is necessary to mitigate the dangers connected with cloud services, including breaches in security can directly impact privacy and vice versa.

The study also shows an adaptive Software Development Lifecycle (SDLC) model that uses predictive analytics to deal with serverless computing problems like cold-start delays. This model improves application performance on many serverless platforms by combining deep neural networks and Long Short-Term Memory (LSTM) algorithms..

The study examines cloud computing technologies and frameworks to understand their efficiency.The goal is to improve cloud service uptake and advance requirements engineering by tackling the unique challenges of cloud environments. The paper concludes with a conversation about possible future research directions improvements in cloud requirements engineering.

II. LITERATURE REVIEW

Research focuses on reducing cold start time in serverless computing to improve speed and responsiveness.This section examines solutions for addressing cold start difficulties, including container management and dynamic optimization of serverless architectures.

To address cold start difficulties, there are two general strategies: minimizing delay and occurrences. Pre-warmed containers and container pooling are two ways for reducing cold start delays. They aim to reduce the time necessary for resource allocation. Xu et al. proposed employing Long Short-Term Memory (LSTM) models to forecast future invocations. This allows for adaptive warm-up tactics that optimize container pre-launching. Akkus et al. used messaging queues to reduce internal event delays, whereas Oakes et al. used Zygote techniques to pre-import libraries, reducing the time spent loading function dependencies into memory.

Container reuse is effective in reducing cold starts in serverless platforms like AWS Lambda, Google Cloud Functions, and Microsoft Azure. Containers are paused after function execution and reused for subsequent requests within a fixed idle-container window. OpenFaaS, Knative, and Fission also offer similar mechanisms, with pools of warm containers that remain available for a period of time.Furthermore, AWS Lambda's provided concurrency feature enables developers to specify the amount of warm containers required for future requests.

Cold start mitigation techniques have been optimized using reinforcement learning approaches. Agarwal et al. used Q-learning to automatically scale function instances depending on resource consumption and request patterns. Shahrad et al. established the Hybrid Histogram Policy, which optimizes idle-container windows based on function characteristics, reducing resource waste and cold start delays.

Machine learning methods were utilized to forecast. function invocations and dynamically change container management, leading to improved adaptability. Xu et al. and Gunasekaran et al. used LSTM networks to predict future workloads and optimize resource management.Furthermore, deep reinforcement learning algorithms have been investigated as viable solutions for cloud environments because of their capacity to adapt to continuous state spaces and fluctuating resource needs..

The OpenWhisk platform suggests using pre-warmed containers, classified by memory and programming language, to minimize cold start latency. This strategy reduces the time necessary to prepare a container because containers are ready ahead of time for various function executions. For example, OpenWhisk normally includes two pre-warmed Node.js containers. When a request is assigned, a pre-warmed container is immediately initialized. This ensures that the system is constantly ready. Process future requests without experiencing severe delays. Although this technique increases response times, its downside is the limited quantity of pre-warmed containers accessible, which could become troublesome during heavy demand.There remains potential to explore more dynamic allocation mechanisms that could scale up as demand increases [2].

Akkus et al. created the SAND platform to simplify container preparation in serverless settings by segregating logic at the application level. This platform decreases overhead associated with function invocations by combining related functions into the same container, lowering container startup time.The SAND platform takes a step further by using lighter-weight isolation methods between functions within the same application, resulting in significantly lower cold start latency. Furthermore, this work is notable for introducing strategies to reduce internal event delays using a message queue. More research is needed to evaluate the SAND platform's effectiveness in real-time production applications, although it appears promise for reducing cold starts.Future work might also investigate whether this approach is suitable for a wider variety of programming languages or cloud environments [1].

The SOCK platform, developed by Oakes et al., aims to improve container-based performance by tackling bottlenecks in the Linux kernel. The solution includes altering how containers handle function calls by employing a technique called Zygote, which pre-imports needed libraries into containers before function invocations. This decreases the time taken to load dependencies during runtime, resulting in considerable speedup. While the SOCK platform has showed promise in lowering cold start time, the implementation of these optimizations raises concerns about how they will interact with other system components, particularly in highly concurrent applications. Future study could investigate more adaptable solutions that automatically modify the quantity of pre-imported libraries based on real-time demands [3].

Silva et al. devised the Prebaking approach, which reuses

Technique	Key Features	Advantages	Disadvantages	Future Work
LSTM for Warm-Up	Predicts future invocations to optimize pre-launching.	Adaptive warm-up tactics reduce delays.	Requires accurate prediction models.	Explore integration with real-time metrics.
Container Pooling	Reuses paused containers for subsequent requests.	Reduces resource allocation time.	Limited warm container availability during high demand.	Investigate dynamic scaling methods.
SAND Platform	Segregates logic to lower invocation overhead.	Decreases cold start latency effectively.	Needs evaluation in production environments.	Assess language compatibility and scalability.
SOCK Platform	Uses Zygote to pre-import libraries before invocations.	Decreases loading times of dependencies.	Interaction with concurrent processes may be complex.	Explore adaptability to changing workload patterns.
Prebaking Method	Reuses snapshots for fast container initialization.	Significantly lowers startup time.	Complexity with multiple libraries and dependencies.	Optimize management for dynamic workloads.
Adaptive Container Pooling (ACPS)	Groups pre-launched containers by programming language.	Enhances resource utilization under variable workloads.	Prediction accuracy for workloads needs improvement.	Implement machine learning for better predictions.
Knative Serving	Dynamic scaling based on expected demand.	Improves resource utilization and reduces cold starts.	Performance validation needed across environments.	Study effectiveness in large-scale applications.
WLEC Architecture	Utilizes warm and template queues for cold start management.	Provides a scalable solution for high request volumes.	Balancing resource use and performance is challenging.	Develop advanced resource allocation strategies.
Function Fusion	Combines multiple functions to reduce initialization.	Minimizes overhead for interconnected workflows.	May have limitations in multi-cloud settings.	Investigate in complex, large-scale systems.
Pagurus	Shares libraries among warm instances for function execution.	Reduces latency by reusing libraries.	Faces challenges with complex dependencies.	Evaluate effectiveness in diverse distributed systems.
HotC Method	Reuses runtime environments without reinitialization.	Enhances efficiency and reduces cold start times.	Adaptive strategies may be needed for variable loads.	Expand to multi-cloud environments and complex workloads.
FaaSLight	Loads only essential code for function execution.	Reduces delays caused by unnecessary code.	Needs assessment in complex systems.	Adapt for additional programming languages.

TABLE I: Comparison of Cold Start Mitigation Techniques in Serverless Computing

container snapshots created during previous function executions. Prebaking uses snapshots to quickly initialize new containers, reducing preparation time.This method was assessed using prototypes and showed a significant reduction in cold start time when compared to typical Unix process launch routines. While Prebaking provides a reliable solution for decreasing startup time, it is interesting investigating how this strategy would function in larger, more complex applications with various libraries and dependencies to handle. Future research could focus on optimizing snapshot management for systems with rapidly changing function needs [4].

Xu et al. suggested an adaptive container pool scaling approach (ACPS) that groups pre-launched containers by programming language. The major innovation is anticipating the number of cold starts for each category and dynamically modifying the number of container instances accordingly.This strategy minimizes cold starts and enhances resource usage, especially under unpredictable workloads. While the ACPS technique is quite effective in reducing startup times, it might be improved to improve prediction accuracy, particularly when dealing with less predictable or more bursty workloads. This strategy can be improved Using real-time monitoring and machine learning to anticipate future function invocation trends [5].

Lin et al. improved container management in serverless architectures using the Knative Serving platform, addressing the cold start issue. Their technique uses an adaptive container pool scaling model to maximize resource utilization by ensuring the appropriate number of containers are always available depending on expected demand.This strategy dramatically decreases the time required to launch containers by categorizing them into pools based on programming languages and other execution requirements. The Knative platform's strength is its ability to dynamically modify the size of the container pool to fit changing traffic. However, more work is needed to examine the model's performance in multiple cloud environments and workloads to validate its robustness in larger real-world scenarios [6].

Solaiman et al. created WLEC, a novel container management architecture that uses warm and template queues to manage cold start delays. A template queue stores a unique copy of each heated container, making it easy to quickly initialize new containers when needed.This design improves on prior methods by providing a more scalable and efficient way to handle the cold start problem, especially when dealing with high-volume request patterns.However, the WLEC paradigm brings the difficulty of balancing resource consumption and performance, especially under changing workloads. Future

Approach	Techniques Used	Accuracy	Complexity	Scalability
LSTM for Warm-Up	Uses LSTM models to predict invocations for adaptive warm-up.	High	Moderate	Moderate
Container Pooling	Reuses paused containers to handle subsequent requests.	Moderate	Low	High
SAND Platform	Segregates application logic to optimize container usage.	Moderate	Moderate	Moderate
SOCK Platform	Employs Zygote for pre-importing libraries.	High	High	Moderate
Prebaking Method	Utilizes snapshots to speed up container initialization.	Moderate	Moderate	Moderate
Adaptive Container Pooling (ACPS)	Groups containers by language to improve resource use.	Moderate	Moderate	High
Knative Serving	Implements adaptive scaling based on traffic demand.	High	Moderate	High
WLEC Architecture	Uses warm and template queues for container management.	Moderate	High	Moderate
Function Fusion	Combines multiple functions into a single container.	Moderate	Moderate	Moderate
Pagurus	Shares libraries among warm instances to reduce latency.	Moderate	Moderate	Moderate
HotC Method	Reuses runtime environments to avoid reinitialization.	High	High	Moderate
FaaSLight	Loads only essential code for function execution.	High	Moderate	High

TABLE II: Comparison of Serverless Computing Approaches

research could examine more advanced approaches to dynamically allocate resources across warm and template queues, better meeting the needs of large-scale serverless applications [7].

Lee et al. developed a way to reduce cold start time in parallel process executions by function fusion.This technique combines numerous functions into a single container, eliminating the overhead of container initialization for each individual function. Their model also includes a reaction time algorithm that changes function execution based on projected latencies, so increasing the total response time of serverless workflows. The solution is especially successful in situations when workflows are heavily intertwined. This technique improves cold start times in some circumstances, but it may need further investigation to understand its limitations when applied to more complex, multi-cloud settings with various workloads [8].

Li et al. developed Pagurus, an innovative architecture that reduces startup time in serverless platforms by allowing containers to share libraries amongst warm instances. This method allows existing containers to ”borrow” libraries from other containers, reducing the time necessary to start a function.This library-sharing approach provides a versatile and resource-efficient solution to handle cold beginnings, especially in systems where comparable functions frequently reuse libraries.While this solution reduces latency, it may face issues when dealing with containers with complex dependencies. More research is needed to evaluate how this technique operates across varied applications and bigger scales in distributed systems ([9]).

Suo et al. presented HotC, a container runtime reuse method that addresses the cold start problem by reusing existing runtime environments. This eliminates the need to restart containers or keep them constantly warmed. This The approach keeps a supply of ready-to-use containers that can be used instantly for new requests. By eliminating the requirement for frequent container reinitialization, HotC dramatically improves serverless platform efficiency and minimizes cold start time. In addition, the model uses a Markov chain-based predictive method to estimate the ideal number of containers to keep warm.Although HotC has shown significant improvements in latency reduction, future research could focus on expanding this strategy to multi-cloud environments or systems with highly variable workloads. Container pooling may require more adaptive strategies to cope with diverse operational conditions [10].

Lee et al. proposed FaaSLight, a solution for reducing cold start times in serverless computing. It uses application-level analysis to load just relevant code for function execution. FaaSLight analyzes the application’s function-level call graph to identify essential and optional code. Loading only essential code at the start reduces delays caused by extraneous code, resulting in faster function execution times. This approach was examined using Python and JavaScript prototypes, resulting in considerable efficiency improvements.While the results are promising, future study could improve FaaSLight’s capabilities by adapting the system for other languages and assessing its performance in complicated and large-scale systems with continually evolving codebases [11].

References

[1] J. Weinman, The future of cloud computing, in: 2011 IEEE Technology Time Machine Symposium on Technologies Beyond 2020, 2011, pp.

1–2, Link. doi:10.1109/TTM.2011.6005157.

[2] D. Baloni, C. Bhatt, S. Kumar, P. Patel, T. Singh, The evolution of virtualization and cloud computing in the modern computer era, Link (2023). doi:10.1109/ICCSAI59793.2023.10421611.

[3] M. Alam, S. Mustajab, M. Shahid, F. Ahmad, Cloud computing: Architecture, vision, challenges, opportunities, and emerging trends, in: 2023 International Conference on Computing, Communication, and Intelligent Systems (ICCCIS), 2023, pp. 829–834, Link. doi:10.1109/ICCCIS60361.2023.10425507.

[4] P. Vahidinia, B. Farahani, F. S. Aliee, Mitigating cold start problem in serverless computing: A reinforcement learning approach, IEEE Internet of Things Journal 10 (5) (2023) 3917–3927, Link. doi:10.1109/JIOT.2022.3165127.

[5] A. C. Zhou, R. Huang, Z. Ke, Y. Li, Y. Wang, R. Mao, Tackling cold start in serverless computing with multi-level container reuse, Link (2024). doi:10.1109/IPDPS57955.2024.00017.

[6] T. Y. Htet, T. Shwe, I. Mendonca, M. Aritsugi, Pre-warming: Alleviating cold start occurrences on cloud-based serverless platforms, Link (2024). doi:10.1109/EdgeCom62867.2024.00018.

[7] T. P. Bac, M. N. Tran, Y. Kim, Serverless computing approach for deploying machine learning applications in edge layer, Link (2022). doi:10.1109/ICOIN53446.2022.9687209.

[8] S. Agarwal, M. A. Rodriguez, R. Buyya, A reinforcement learning approach to reduce serverless function cold start frequency, Link (2021). doi:10.1109/CCGrid51090.2021.00097.

[9] Y. C. Zhou, X. P. Liu, X. N. Wang, L. Xue, X. X. Liang, S. Liang, Business process centric platform-as-a-service model and technologies for cloud enabled industry solutions, Link (2010). doi:10.1109/CLOUD.2010.52.

[10] M. Ribas, A. Sampaio Lima, J. Neuman de Souza, F. Rubens de Carvalho Sousa, L. Oliveira Moreira, A platform as a service billing model for cloud computing management approaches, Link (2016). doi:10.1109/TLA.2016.7430089.

[11] R. Seethamraju, Adoption of saas enterprise systems — a comparison of indian and australian smes, Link (2014). doi:10.1109/ICTER.2014.7083898.