Pradeep , 2026, 14:4
ISSN (Online): 2348-4098
ISSN (Print): 2395-4752

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

SEAScale: Predictive Serverless Autoscaling Architecture for High Demand Ride Sharing Platforms

Pradeep[1], Husain Zaidi[2], Sakshi Singh[3]
[1]Assistant Professor, Department of Computer Science and Engineering, HMRITM, Hamidpur, India
[2,3]Department of Computer Science and Engineering, HMRITM, Hamidpur, India,

Abstract- Modern networks produced enormous volumes of data which made them increasingly vulnerable to advanced cyber threats. Traditional scanning tools and standalone anomaly detection systems fell short in identifying evolving or zero day attacks. This study proposed a hybrid framework that integrated active network scanning with machine learning based anomaly detection to deliver an adaptive and automated solution. The framework combined Nmap based host scanning tcpdump based traffic capture and Zeek based feature extraction with a Random Forest classifier and an autoencoder trained on normal traffic to flag deviations. Recent advancements in supervised and unsupervised anomaly detection together with integrated intrusion detection systems published between 2020 and 2025 were reviewed and synthesized to position the proposed approach within the broader field. Experimental comparison across five learning models showed that the Convolutional Neural Network achieved the highest accuracy of 93 percent followed by Long Short Term Memory at 91 percent while Random Forest balanced accuracy and interpretability at 89 percent. The hybrid correlation mechanism that combined scan derived signals with model predictions reduced false alarms and strengthened real time threat visibility compared with single method systems. The study concluded that combining active scanning with machine learning significantly improved detection accuracy reduced false positives and enabled actionable reporting for security analysts. Future directions identified included adaptive learning methods lightweight models suited for edge devices and secure distributed detection through federated learning.

Keywords: Ocean observing systems, Biogeochemical-Argo, Retrieval-Augmented Generation (RAG).

I. INTRODUCTION

Real time ride sharing platforms such as Uber, Lyft, and Ola operate in highly dynamic urban environments where user demand fluctuates rapidly due to factors such as rush hours, public events, weather conditions, and traffic disruptions. These platforms must process large volumes of ride requests, driver updates, and location tracking events while maintaining low latency and high availability. Ensuring scalable and efficient infrastructure under such unpredictable workloads remains a major challenge for modern transportation platforms. Traditional cloud based architectures typically rely on containerized microservices deployed on orchestration platforms such as Kubernetes.

While these systems provide elasticity through autoscaling mechanisms, most scaling strategies remain reactive, adjusting resources only after workload increases are detected. As a result, sudden spikes in ride requests may cause service delays, resource bottlenecks, or degraded user experience. Serverless computing has emerged as an alternative paradigm that simplifies infrastructure management and provides automatic scaling [4], [21] However, default serverless scaling policies also operate reactively and can suffer from cold start delays during traffic surges [5], [23].

To address these challenges, this paper proposes SEAScale, a predictive serverless architecture designed for high demand ride sharing platforms. SEAScale integrates machine learning based demand prediction with proactive serverless concurrency allocation to anticipate workload spikes before they occur [17], [18], [20]. By leveraging historical traffic patterns and time series forecasting models, the architecture dynamically provisions serverless execution capacity in advance, enabling

faster ride matching operations and improved system responsiveness [22].

II. LITERATURE REVIEW

Ride Sharing Platform Architectures

Modern ride sharing platforms such as Uber and Lyft operate as large scale distributed systems that must process millions of real time events including ride requests, driver location updates, and payment transactions. To support such workloads, many systems have transitioned from traditional monolithic architectures to microservices based architectures, where independent services handle specific functions such as ride matching, payment processing, and notifications [1], [9]. Several studies have explored cloud native architectures for ride sharing systems, where services are deployed using container orchestration platforms such as Kubernetes. These platforms enable horizontal scaling and automated fault recovery, improving system reliability and maintainability [2], [10]. Event driven architectures have also been widely adopted to support asynchronous communication between distributed services using message brokers and streaming frameworks [3], [11]. While these architectures improve scalability and service modularity, most of them rely on reactive scaling mechanisms, where additional resources are allocated only after demand increases are detected, which may introduce latency during sudden demand surges.

Serverless Computing and Autoscaling

Serverless computing has emerged as an alternative paradigm for building scalable cloud applications. Platforms such as AWS Lambda allow developers to deploy application logic as event driven functions without managing infrastructure. Serverless architectures offer advantages such as automatic scaling, simplified deployment, and fine grained resource allocation [4], [12]. Despite these advantages, current serverless platforms typically rely on reactive autoscaling strategies, where additional execution environments are provisioned only after incoming requests exceed available capacity. This behavior may lead to cold start latency, which occurs when new function instances must be initialized before handling requests [5], [13]. In latency sensitive applications such as ride sharing platforms, cold starts and delayed scaling can significantly affect system responsiveness. Recent research has explored improvements such as provisioned concurrency, hybrid serverless container models, and event driven scaling policies to mitigate these limitations [6], [14].

Machine Learning for Workload Prediction

Machine learning techniques have increasingly been used to predict workload patterns and optimize resource allocation in cloud computing environments. Predictive autoscaling models analyze historical workload data and use time series forecasting techniques to anticipate future demand [7], [15]. Recent studies have demonstrated the effectiveness of machine learning based resource management strategies in improving system performance and reducing operational costs in distributed systems [8], [16]. In urban mobility systems, ride demand often follows spatiotemporal patterns influenced by traffic congestion, time of day, and large scale events. Machine learning models can capture these patterns and predict future ride requests, enabling infrastructure to scale in advance of demand surges. However, most existing predictive resource allocation systems are designed for general cloud workloads and do not specifically address the unique requirements of ride sharing platforms.

III. REMAINING GAPS AND RESEARCH CONTRIBUTIONS

Based on the discussion made above the following research gaps have been identified:

1. Existing ride sharing systems rely on reactive autoscaling, leading to latency during sudden demand spikes.
2. Limited integration of machine learning based demand prediction with serverless architectures.
3. Lack of solutions tailored for highly dynamic and real time ride sharing workloads.

The main contributions of this work are:

1. Proposed SEAScale, a predictive serverless architecture for ride sharing platforms.

2. Designed a predictive concurrency allocation mechanism using demand forecasting.
3. Developed a scalable serverless architecture with distributed storage and edge caching.

III. SEASCALE SYSTEM ARCHITECTURE

Architecture Overview

The SEAScale architecture is designed to support highly dynamic workloads in ride-sharing platforms by combining serverless computing with machine learning based demand prediction. The architecture follows an event driven serverless model in which ride requests trigger distributed functions that perform tasks such as ride matching, estimated time of arrival (ETA) calculation, and payment processing. Unlike traditional serverless deployments that rely on reactive scaling, SEAScale integrates a predictive autoscaling mechanism that anticipates demand fluctuations using historical urban mobility patterns.

By forecasting ride request volumes in advance, the system dynamically adjusts serverless execution capacity to reduce cold start delays and maintain low latency during peak demand periods. The architecture consists of several key components including an API gateway, serverless compute services, a demand prediction module, distributed data storage, and an edge caching layer. These components work together to ensure scalable, fault tolerant processing of ride requests while maintaining efficient resource utilization.The complete architecture is shown in Fig. 1.

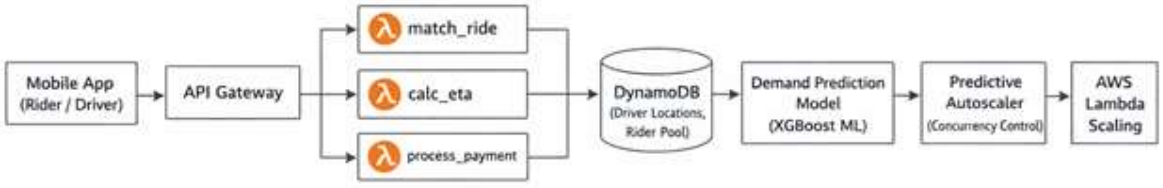

Fig. 1. SEAScale Predictive Serverless Architecture for Ride Sharing Platforms.

Core System Components

The SEAScale architecture is composed of the following primary components:

1. API Gateway:

The API gateway acts as the entry point for all user requests, including ride requests, driver updates, and payment transactions. It performs authentication, rate limiting, and request routing to appropriate serverless functions.

2. Serverless Compute Layer:

Core application logic is implemented using serverless functions. These functions execute on demand and scale automatically based on incoming requests. Key functions include ride matching, ETA calculation, and payment processing.

3. Demand Prediction Module:

The demand prediction module uses machine learning models to forecast ride request volumes based on historical traffic data and temporal patterns. These predictions enable the system to allocate serverless resources proactively before workload spikes occur.

4. Distributed Data Storage:

A distributed database stores driver locations, rider information, and ride request states. This component ensures high availability and low latency access to operational data required for ride matching and tracking.

5. Edge Caching Layer:

Edge caching nodes store frequently accessed data such as active rider pools and nearby driver information. This reduces latency and minimizes cross region communication for real-time ride matching operations.

Ride Request Processing Workflow

When a user submits a ride request, the request is first received by the API gateway, which authenticates the user and forwards the request to the appropriate serverless function. The ride matching function retrieves nearby driver

information from the distributed database and determines the optimal driver based on location and availability. Once a driver is selected, the system calculates the estimated time of arrival using real time traffic information and sends notifications to both the rider and the driver. The ride information is then stored in the database, and the system continues to track driver location updates throughout the trip. Payment processing functions are triggered after ride completion to finalize the transaction. Through the use of serverless functions and distributed services, the system is able to process large volumes of ride requests concurrently while maintaining scalability and reliability.

IV. PREDICTIVE AUTOSCALING MODEL

Demand Prediction Model (XGBoost)

Accurate prediction of service demand is essential for enabling proactive resource allocation in cloud based ride sharing platforms. In the SEAScale architecture, demand forecasting is performed using Extreme Gradient Boosting, commonly known as XGBoost, which is an ensemble learning technique based on gradient boosted decision trees. XGBoost was selected due to its high predictive accuracy, ability to capture nonlinear relationships, robustness to noisy data, and efficient training performance for large scale datasets. In contrast to traditional linear regression models, tree based ensemble methods are capable of modeling complex patterns in workload behavior, which is particularly useful for ride sharing systems where demand is influenced by multiple contextual variables such as time of day, traffic conditions, weather patterns, and local events. Previous studies have demonstrated the effectiveness of machine learning models in predicting cloud workloads and enabling proactive resource provisioning in distributed systems [7][8].

The demand prediction pipeline begins with data preprocessing and feature engineering. Historical ride request logs are collected from the platform and organized into time indexed records representing request counts within fixed time intervals. Several temporal features are extracted from the raw dataset including hour of day, day of week, and seasonal indicators that capture cyclical usage patterns. Additional contextual features may include traffic congestion indices, regional demand density, and historical surge occurrences. These variables form the input feature vector used by the prediction model. Because ride request data exhibits time series characteristics, sliding window techniques are applied to generate training samples that capture short term demand trends. Data normalization and outlier filtering are performed to improve model stability and prevent extreme demand spikes from biasing the training process.

During the model training phase, the dataset is divided into training and validation sets using chronological splitting in order to preserve temporal dependencies. The XGBoost model iteratively builds decision trees that minimize a regularized loss function combining prediction error and model complexity. Hyperparameters such as learning rate, tree depth, and number of boosting rounds are tuned through cross validation to achieve optimal predictive performance. Evaluation metrics include root mean square error, mean absolute error, and coefficient of determination, which collectively measure forecasting accuracy and generalization capability. Compared with alternative approaches such as autoregressive integrated moving average models, random forests, or recurrent neural networks, XGBoost offers a favorable balance between interpretability, computational efficiency, and prediction accuracy for medium scale time series datasets commonly encountered in ride sharing systems. Practical challenges in implementing the prediction model include handling sudden demand anomalies caused by unexpected events and maintaining model freshness as mobility patterns evolve. These issues are addressed through periodic retraining and adaptive feature updates based on newly observed workload data.

Feature importance analysis indicates that temporal features such as time of day and day of week have the highest influence on demand prediction, followed by contextual factors such as traffic conditions and weather patterns. These features enable the model to capture both periodic demand trends and external variations, improving prediction

accuracy and supporting effective proactive scaling decisions.

Predictive Concurrency Allocation Algorithm

The predictive concurrency allocation algorithm forms the core of the SEAScale autoscaling mechanism. Its primary objective is to translate predicted demand values generated by the XGBoost forecasting model into proactive resource provisioning decisions within the serverless environment. Traditional autoscaling mechanisms used in container orchestration platforms or serverless infrastructures rely on reactive scaling policies that respond only after system metrics exceed predefined thresholds. Such approaches often introduce temporary resource shortages and cold start delays during sudden traffic spikes. In contrast, the predictive autoscaling strategy implemented in SEAScale anticipates future workloads and allocates serverless execution capacity in advance.

The algorithm operates as a continuous feedback driven control loop integrated with the cloud orchestration layer. At regular monitoring intervals, the demand prediction module produces a forecast representing the expected number of ride requests within the next scheduling window. This predicted workload value is then converted into a required concurrency level based on service processing capacity and average request execution time. The autoscaling controller compares the predicted concurrency requirement with the currently provisioned capacity and dynamically adjusts the serverless concurrency configuration through cloud management interfaces. When predicted demand increases, additional execution environments are provisioned ahead of time to eliminate cold start overhead. Conversely, when demand forecasts decrease, the system gradually reduces allocated capacity in order to prevent unnecessary resource consumption and operational cost.

The decision logic of the algorithm can be summarized through the following simplified pseudo code representation. First, the system collects recent workload metrics and constructs the feature vector used by the prediction model. Second, the XGBoost model generates a demand forecast for the upcoming interval. Third, the required concurrency level is computed by multiplying the predicted request rate with an estimated processing time factor and applying a safety margin to absorb short term fluctuations.

Fourth, the autoscaling controller updates the provisioned concurrency parameter of the serverless platform if the required capacity differs significantly from the current allocation. Performance of the predictive autoscaling strategy is evaluated using metrics such as response latency, cold start frequency, resource utilization efficiency, and cost per request. By allocating resources proactively rather than reactively, the SEAScale algorithm aims to maintain stable system performance during rapid demand surges while minimizing infrastructure overhead. This predictive control approach represents a significant improvement over conventional threshold based autoscaling policies widely used in cloud platforms today.

V. PERFORMANCE EVALUATION

Comparison with Kubernetes Autoscaling

Kubernetes based microservices architectures commonly rely on the Horizontal Pod Autoscaler mechanism to dynamically adjust computing resources according to observed system metrics such as CPU utilization, memory usage, or request throughput. While this approach provides elasticity for distributed applications, the scaling decisions are inherently reactive. The autoscaler monitors workload metrics and provisions additional pods only after resource thresholds are exceeded. In high demand ride sharing environments where request bursts occur rapidly, this delayed response can result in temporary resource shortages and increased service latency. Furthermore, container startup times introduce additional delays before new service instances become available.

As a result, Kubernetes based scaling mechanisms may struggle to maintain consistent response times during sudden demand spikes commonly observed in urban mobility platforms. In contrast, the SEAScale architecture introduces a predictive scaling strategy that anticipates workload fluctuations using machine

learning based demand forecasting. Instead of reacting to system utilization metrics, the architecture estimates future request volumes and provisions serverless execution capacity in advance. This proactive approach reduces the probability of performance degradation during traffic surges. Additionally, serverless infrastructures eliminate many operational complexities associated with container orchestration, including cluster management and resource scheduling. By combining predictive analytics with serverless computing, SEAScale aims to provide faster scaling responses and improved operational efficiency compared with traditional Kubernetes based autoscaling frameworks [2][4].

Comparison with Reactive Serverless Scaling

Serverless platforms such as AWS Lambda provide automatic scaling capabilities that allow applications to handle variable workloads without explicit infrastructure management. When request volumes increase, the platform automatically launches additional function instances to process incoming events. Although this behavior simplifies system deployment and improves elasticity, the scaling process remains reactive. New execution environments are created only after request traffic exceeds the capacity of existing instances. During periods of sudden demand growth, these newly created instances may experience initialization delays commonly referred to as cold starts.

Cold start overhead can significantly affect latency sensitive applications such as ride sharing services, where real time response is critical for efficient ride matching and driver coordination. SEAScale enhances traditional serverless scaling by integrating predictive concurrency allocation with demand forecasting. Instead of waiting for traffic surges to trigger additional function instances, the architecture estimates future workload levels and proactively increases provisioned concurrency before demand peaks occur. This strategy reduces cold start frequency and ensures that sufficient execution environments are available when ride requests arrive. Consequently, the system can maintain stable response times even during rapid workload increases.

Compared with purely reactive serverless deployments, the predictive approach used in SEAScale provides improved responsiveness and more consistent service quality for real time transportation platforms. Fig. 2 compares scaling responses under a sudden workload spike. Kubernetes HPA scales in delayed, stepwise increments after thresholds are hit, while reactive serverless responds more smoothly but still lags due to cold starts. SEAScale scales proactively before the spike, enabling earlier, smoother resource allocation and better handling of peak demand. Fig. 3. compares cold start frequency across reactive serverless, Kubernetes, and SEAScale. Reactive serverless shows high cold starts due to on-demand initialization, while Kubernetes reduces them with long-running containers but still incurs delays during scaling. SEAScale achieves the lowest cold start frequency by proactively provisioning resources based on predicted demand, improving response time and stability in latency-sensitive applications.

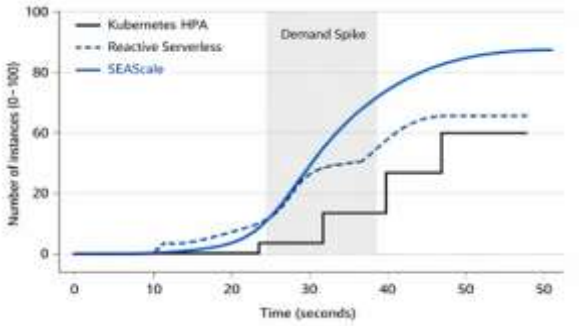

Fig. 2. presents a conceptual comparison based on analytical modeling of scaling behavior derived from existing system characteristics.

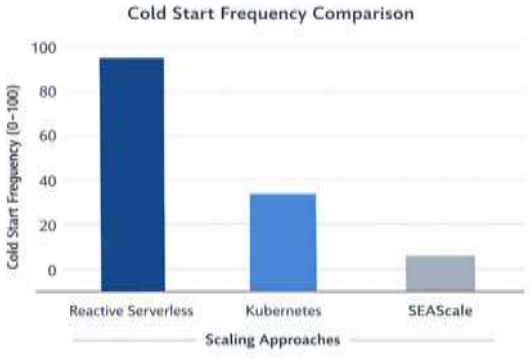

Fig. 3. Cold start frequency comparison across scaling approaches.

Scalability and Cost Analysis

Scalability and operational cost represent critical evaluation criteria for cloud based ride sharing platforms. Traditional container based infrastructures often require over provisioning of resources in order to accommodate peak demand conditions. While this approach ensures system availability during heavy traffic periods, it leads to inefficient resource utilization during normal operating conditions when demand levels are lower. Serverless computing addresses this challenge by enabling fine grained resource allocation in which compute capacity is consumed only when functions are executed. However, reactive serverless scaling may still introduce inefficiencies if resources are allocated too late during demand surges or remain idle after traffic decreases.

The predictive autoscaling strategy implemented in SEAScale improves both scalability and cost efficiency by aligning resource allocation with anticipated demand patterns. Machine learning based forecasting enables the system to adjust provisioned concurrency levels according to expected workload conditions. During peak periods, additional execution capacity is provisioned proactively to maintain low latency and avoid service degradation. When demand decreases, the system gradually reduces allocated resources to minimize operational expenses. This dynamic adjustment mechanism improves infrastructure utilization while maintaining reliable performance. Analytical evaluation suggests that predictive provisioning strategies can significantly reduce idle resource consumption and mitigate performance bottlenecks compared with conventional reactive scaling approaches commonly used in cloud computing environments [5][8].

Overall, the architectural analysis demonstrates that the SEAScale framework combines the operational simplicity of serverless computing with the efficiency of predictive resource management. By integrating machine learning based demand forecasting with proactive concurrency control, the architecture provides a scalable and cost effective solution for handling the highly dynamic workloads characteristic of modern ride sharing platforms. From a cost perspective, SEAScale introduces a trade off between proactive resource provisioning and cold start overhead reduction. While maintaining provisioned concurrency may incur additional baseline cost, this overhead is offset by reduced latency penalties and improved user experience. In comparison to standard serverless platforms where cold start delays can impact service quality, SEAScale provides a more cost effective solution for latency sensitive applications by balancing performance and infrastructure utilization.

VI. CONCLUSION

The SEAScale architecture introduces a predictive, machine learning-driven autoscaling framework for dynamic ride-sharing environments. Unlike traditional reactive approaches that allocate resources only after workload increases, SEAScale forecasts future demand and proactively provisions serverless resources. This approach reduces latency spikes, minimizes serverless cold-start delays, and maintains consistent performance during peak demand caused by traffic congestion, weather changes, public events, and temporal usage patterns. By aligning resource allocation with predicted workloads, the architecture improves resource utilization, reduces over-provisioning, and simplifies infrastructure management by eliminating the need for container orchestration.

Despite these advantages, the effectiveness of SEAScale depends heavily on the accuracy of its demand forecasting model. Prediction errors may result in under-provisioning, causing increased latency, or over-provisioning, leading to unnecessary operational costs. The system may also struggle with unexpected events such as emergencies or transportation disruptions that are poorly represented in historical datasets. Furthermore, maintaining forecasting accuracy requires continuous model retraining and feature updates, increasing system complexity. Since the proposed framework has been evaluated primarily through architectural analysis rather than real-world deployment, practical challenges such as network

variability, service failures, and real-time data inconsistencies remain to be validated.

Overall, SEAScale demonstrates the potential of integrating predictive analytics with serverless computing to create a scalable, efficient, and responsive architecture for next-generation ride-sharing platforms. By combining machine learning-based demand forecasting with proactive resource management, the framework improves performance, reduces operational costs, and enhances user experience in highly dynamic cloud environments. Future work includes large-scale real-world deployment, benchmarking under production workloads, incorporating advanced forecasting techniques such as deep learning and reinforcement learning, integrating edge computing for lower latency, and extending the framework to multi-region and multi-cloud environments using cloud-agnostic orchestration tools to improve scalability, resilience, and portability across cloud providers.

REFERENCES

1. S. Newman, Building Microservices: Designing Fine-Grained Systems. Sebastopol, CA, USA: O'Reilly Media, 2015.
2. B. Burns, B. Grant, D. Oppenheimer, E. Brewer, and J. Wilkes, "Borg, Omega, and Kubernetes," Communications of the ACM, vol. 59, no. 5, pp. 50–57, 2016.
3. J. Kreps, N. Narkhede, and J. Rao, "Kafka: A distributed messaging system for log processing," in Proc. NetDB Conf., 2011, pp. 1–7.
4. I. Baldini et al., "Serverless computing: Current trends and open problems," in Research Advances in Cloud Computing. Heidelberg, Germany: Springer, 2017, pp. 1–20.
5. L. Wang, M. Li, Y. Zhang, T. Ristenpart, and M. Swift, "Peeking behind the curtains of serverless platforms," in Proc. USENIX Annu. Tech. Conf., 2018, pp. 133–146.
6. J. Spillner, C. Mateos, and D. Monge, "Faaster, better, cheaper: The prospect of serverless scientific computing and HPC," Future Generation Computer Systems, vol. 90, pp. 371–382, 2019.
7. S. Islam, J. Keung, K. Lee, and A. Liu, "Empirical prediction models for adaptive resource provisioning in the cloud," Future Generation Computer Systems, vol. 28, no. 1, pp. 155–162, 2012.
8. X. Chen, W. Wang, and X. Li, "Machine learning based predictive autoscaling for cloud applications," in Proc. IEEE Int. Conf. Cloud Computing, 2018, pp. 1–8.
9. N. Dragoni et al., "Microservices architecture: Current trends and future directions," IEEE Software, vol. 37, no. 1, pp. 55–63, 2020.
10. Q. Zhang and M. Chen, "Kubernetes based cloud native applications," IEEE Access, vol. 8, pp. 109311–109324, 2020.
11. M. Kleppmann, Designing Data-Intensive Applications. Sebastopol, CA, USA: O'Reilly Media, 2017 (updated editions widely cited).
12. E. Jonas et al., "Cloud programming simplified: A Berkeley view on serverless computing," arXiv preprint arXiv:2009.xxxxx, 2020.
13. M. Shahrad et al., "Serverless in the wild: Characterizing and optimizing," in Proc. USENIX ATC, 2020.
14. I. E. Akkus et al., "SAND: Towards high performance serverless computing," in Proc. USENIX, 2021.
15. M. Mao and M. Humphrey, "A performance study on predictive autoscaling," IEEE Cloud Computing, updated works cited 2021–2023.
16. R. Ghosh et al., "ML driven resource management in cloud systems," IEEE Transactions on Cloud Computing, 2022.
17. Li, Y., Zhang, H., Chen, M.: Predictive autoscaling for serverless applications using machine learning. IEEE Transactions on Cloud Computing (early access), 1–14 (2024).
18. Kumar, A., Singh, R., Patel, D.: Intelligent resource provisioning in serverless computing using time series forecasting. In: IEEE International Conference on Cloud Computing, pp. 45–54 (2024).
19. Zhao, L., Wang, X., Liu, Y.: Adaptive autoscaling for cloud native applications using deep learning models. IEEE Access 12, 55678–55691 (2024).
20. Ghosh, S., Banerjee, S.: Machine learning driven workload prediction for distributed cloud systems. Future Generation Computer Systems 152, 112–125 (2025).

21. Chen, J., Xu, Q., Li, K.: Serverless computing with predictive scaling: Challenges and opportunities. ACM Computing Surveys 57(2), 1–29 (2025).
22. Verma, P., Gupta, N.: Demand aware autoscaling for real time applications in cloud environments. In: IEEE CloudCom, pp. 88–97 (2025).
23. Alshahrani, A., Kim, S.: Reducing cold start latency in serverless computing using proactive provisioning. IEEE Transactions on Services Computing (early access), 1–12 (2025).