Journal of Advanced College of Engineering and Management

AI-Driven Intelligent Auto-Scaling for Cloud Resource Optimization

[1]Sudip Poudel, [2]Kushal Sharma Marasini, [1]Laxmi Prasad Bhatt, [3]Daya Sagar Baral

[1]*MSc. in Computer System and Knowledge Engineering Pulchowk Campus, Institute of Engineering, Tribhuvan University Lalitpur, Nepal*

[2]*MSc. in Information and Communication Engineering Pulchowk Campus, Institute of Engineering, Tribhuvan University Lalitpur, Nepal*

[3]*Asst. Professor Pulchowk Campus, Institute of Engineering, Tribhuvan University Lalitpur, Nepal*

Email: iamsudip91@gmail.com, kmarasini12@gmail.com, lpbhatta0828@gmail.com, dsbaral@pcampus.edu.np

DOI: 10.3126/jacem.v11i1.84521

Abstract

This study introduces a predictive, AI-powered auto-scaling framework designed to optimize resource usage in cloud environments, specifically within Amazon Web Services (AWS). Conventional rule-based scaling methods often result in inefficiencies, either wasting resources or degrading performance. To overcome these challenges, this work employs Long Short-Term Memory (LSTM) neural networks that analyze historical performance data collected from AWS CloudWatch. The system forecasts resource demand trends for EC2 and RDS instances and automates scaling actions using the Boto3 SDK. It evaluates multiple metrics—including CPU usage, memory availability, disk I/O, and network traffic—to make accurate, real-time decisions. Operating in a continuous loop, the model updates hourly to adapt to changing workloads. Experimental evaluation confirms that the proposed approach reduces operational costs and enhances performance reliability. This research delivers a scalable, intelligent solution for cloud resource management, suitable for dynamic application environments where responsiveness and efficiency are critical.

Keywords*—AWS, EC2, LSTM, RDS, SDK*

1. INTRODUCTION

The rapid growth of cloud computing has transformed how organizations deploy and manage digital infrastructure. Services such as Amazon Web Services (AWS) allow users to scale computing resources based on demand, offering flexibility and cost-efficiency. However, traditional auto-scaling methods—primarily based on static thresholds—often fail to adapt to fluctuating workloads, resulting in either resource wastage or degraded performance.

This research is motivated by the need for a smarter, adaptive system that can respond to changing resource demands in real time. Existing reactive models typically trigger scaling decisions only after performance issues have occurred, lacking foresight into upcoming load patterns. By integrating machine learning into the auto-scaling process, this study aims to create a predictive mechanism that anticipates resource requirements before bottlenecks arise.

The purpose of this work is to design and implement an AI-driven auto-scaling system using Long Short-Term Memory (LSTM) neural networks. The goal is to improve resource utilization efficiency, reduce cloud infrastructure costs, and maintain optimal application performance in dynamic environments.
The specific objectives of this research are as follows:

- To design and implement an AI-driven auto-scaling system using LSTM neural networks for AWS EC2 and RDS instances.
- To optimize cloud resource utilization by predicting workload patterns and adjusting resources proactively.
- To compare the performance of the proposed predictive model with traditional threshold-based auto-scaling approaches.
- To validate the system using real AWS CloudWatch metric data and assess cost savings, performance improvements, and scalability.

The key contributions of this research include:

- Development of a predictive auto-scaling framework using LSTM neural networks, capable of handling multi-metric inputs.
- Integration with AWS CloudWatch and Boto3 SDK for real-time metric monitoring and automated scaling.
- Real-data validation of the proposed approach using actual EC2 and RDS instances over a two-month period.
- Demonstrated improvements in both cost optimization and performance stability compared to existing rule-based systems.

2. LITERATURE REVIEW

This thesis advances the field of cloud resource management by addressing several limitations observed in prior research. Studies such as [3] and [4] primarily analyze traditional auto-scaling strategies, focusing on how metric selection and resource pool configurations affect performance. While informative, these works rely heavily on reactive methodologies. In contrast, our approach incorporates predictive analytics using Long Short-Term Memory (LSTM) models, enabling forward-looking, real-time workload forecasting and automated scaling decisions, which adds a proactive layer to resource orchestration. Research works like [6], [1], and [7] explore reinforcement learning-based mechanisms, often within microservices or serverless architectures. While these efforts contribute to adaptive scaling, they are generally confined to narrow environments. Our study

expands the scope to encompass Kubernetes-based containerized infrastructures, integrating multi-metric monitoring to assist more granular, dynamic scaling decisions across varied cloud workloads. In comparison, Paper [5] concentrates on function-level scaling within serverless systems, and [8] emphasizes selecting relevant metrics rather than predictive control. Our research, by employing LSTM-based forecasting across CPU, memory, and I/O metrics, delivers a comprehensive model for intelligent, metric-aware auto-scaling suited to dynamic and heterogeneous workloads. Paper [9] discusses container orchestration using Kubernetes but does not incorporate forecasting mechanisms or real-time metric prediction. Likewise, [10] presents task-level time-series forecasting, yet it lacks integration with a full-stack auto-scaling framework, particularly in multi-tenant or production-grade environments. Overall, most prior literature emphasizes reactive scaling or targets energy efficiency and cost reduction from the consumer's perspective. This thesis presents a more holistic and deployable solution by merging predictive modeling, Kubernetesnative auto-scaling, and AWS-based infrastructure integration. It not only anticipates resource needs but also responds adaptively, offering a scalable, cost-efficient, and intelligent alternative to traditional methods.

3. METHODOLOGY

This section outlines the step-by-step process for developing and deploying the intelligent auto-scaling system for cloud resource optimization. The goal is to design a repeatable framework that enables predictive scaling of AWS EC2 and RDS instances based on real-time metric forecasting.

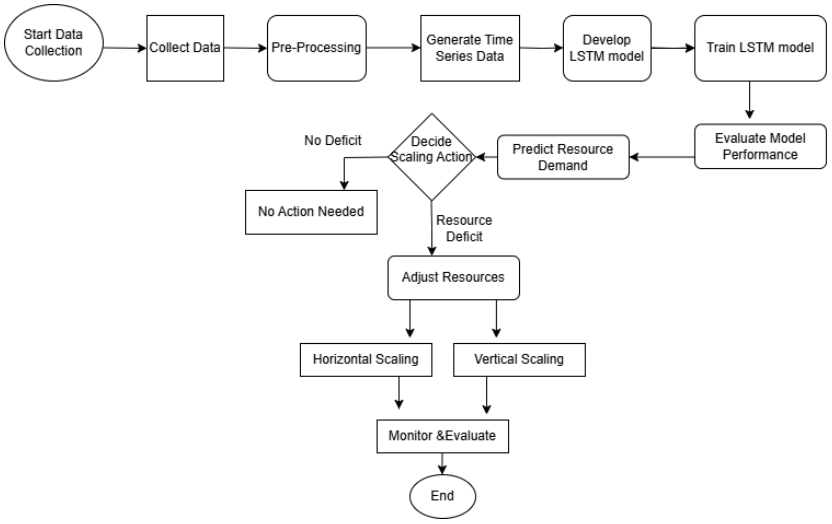

Figure 3.1 Methodology

3.1 System Overview

The system integrates AWS CloudWatch for metric monitoring, Long Short-Term Memory (LSTM) neural networks for forecasting, and AWS Boto3 SDK for scaling operations. It

continuously collects resource metrics, preprocesses them, predicts future workload patterns, and performs auto-scaling actions accordingly.

3.2 Data Collection

Metric data was collected from AWS CloudWatch at 15-minute intervals over a two-month period.

EC2 Metrics Collected:

- CPUUtilization
- NetworkIn
- NetworkOut
- DiskReadOps
- DiskWriteOps
- StatusCheckFailed

RDS Metrics Collected:

- CPUUtilization
- FreeableMemory
- DatabaseConnections
- FreeStorageSpace
- ReadIOPS
- WriteIOPS

These metrics were gathered using Python's Boto3 SDK, and stored in Pandas DataFrames for further processing.

3.3 Data Preprocessing

The raw data was normalized using Min-Max scaling to ensure consistent input for the LSTM model. Missing or incomplete data points were filled with zero or the most recent valid value. Input sequences were reshaped into three-dimensional arrays with the structure (samples, time steps, features), as required by LSTM architecture.

3.4 Model Architecture

Separate LSTM models were developed for EC2 and RDS instances. Each model includes:

- One LSTM layer with 50 hidden units
- ReLU activation function
- 20% Dropout layer for regularization
- Dense output layer with 3 neurons representing possible actions: downgrade, no change, upgrade
- Softmax activation on the output for classification

3.5 Model Training

Models were trained with a batch size of 16 over a maximum of 50 epochs using the Adam optimizer and Mean Squared Error (MSE) as the loss function. Early stopping was applied if validation loss did not improve over 3 consecutive epochs. Data was split chronologically into training and validation sets in an 80:20 ratio to prevent leakage.

3.6 Prediction and Decision Logic

The trained models were used to forecast near-future values of the metrics. Based on the predictions:

For EC2:

- Downgrade if CPU < 20%, NetworkIn < 100MB, and DiskReadOps < 0.5
- Upgrade if CPU > 80%, NetworkIn > 200MB, and DiskWriteOps > 0.5
- No Change if within threshold

For RDS:

- Downgrade if CPU < 20% and connections < 10
- Upgrade if CPU > 80% or ReadIOPS > 100
- No Change if within threshold

3.7 Scaling Execution

Scaling actions were automated via the AWS Boto3 SDK:

EC2 Scaling:

- Stop instance → Modify type → Restart instance
- Bound by allowed instance type range (e.g., t3.nano to t3.medium)

RDS Scaling:

- Modify instance class using ModifyDBInstance with ApplyImmediately=True
- Scaling results in brief downtime; no action taken if at size limit

3.8 Automation Loop

The system runs as a looped script, repeating every hour:

- Fetch latest metrics
- Preprocess and normalize
- Predict next usage levels
- Apply scaling decision
- Sleep for 1 hour

This methodology ensures an automated, real-time, predictive scaling system that adjusts AWS resources dynamically based on workload forecasts, improving both efficiency and performance.

4. RESULTS AND DISCUSSION

The intelligent auto-scaling system developed in this research was deployed and tested on AWS infrastructure using real EC2 and RDS instances. The goal was to assess the system's ability to accurately predict workload patterns and make effective scaling decisions based on forecasted metrics. The evaluation focused on prediction accuracy, responsiveness, cost savings, and comparison with traditional scaling methods.

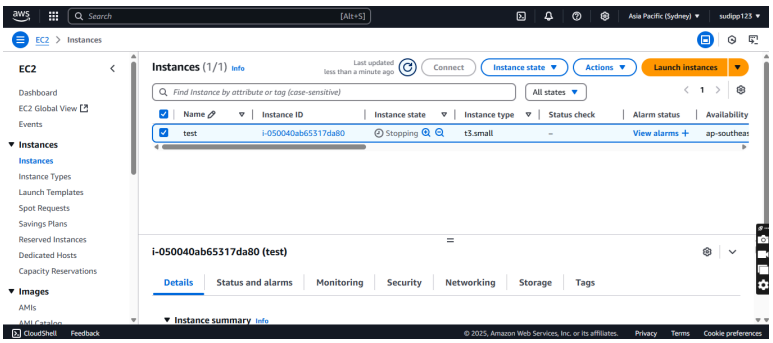

Figure 4.1: EC2 changing

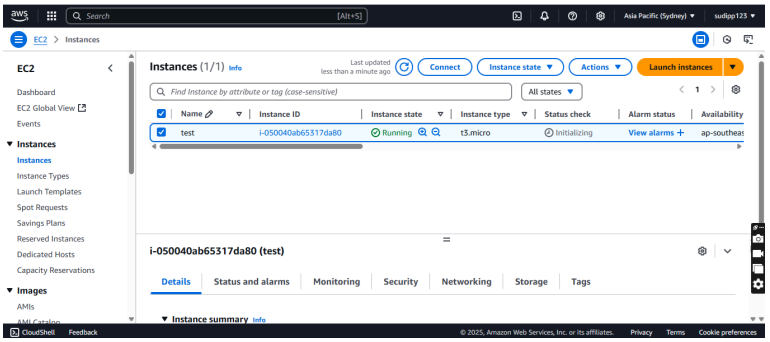

Figure 4.2: EC2 changed

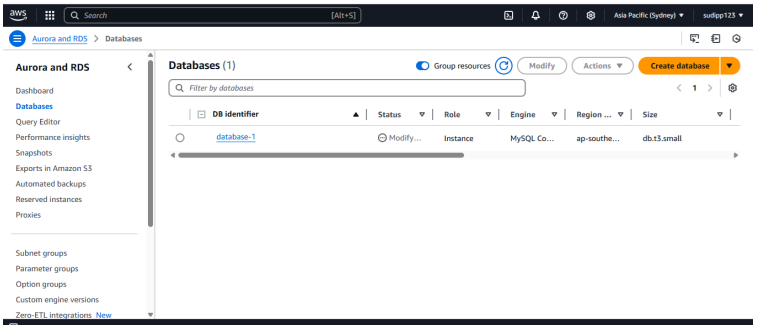

Figure 4.3: RDS changing

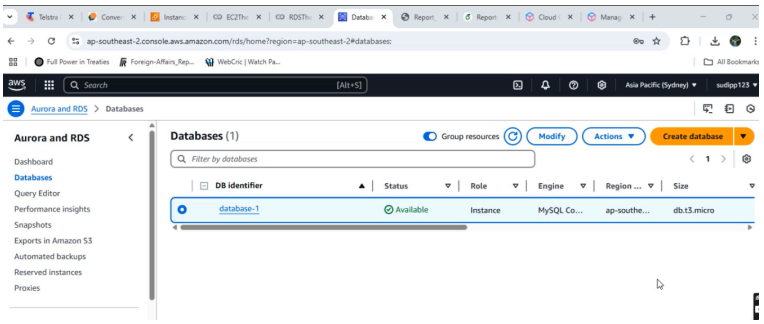

Figure 4.4: RDS changed

Real Data Validation

Unlike many existing studies that rely primarily on simulations or synthetic workloads, this research validates the proposed framework using real historical metric data from AWS CloudWatch. The dataset includes CPU utilization, network activity, disk operations, and database connections collected over a continuous two-month period. The use of real-world data ensures that the proposed system is not only theoretically sound but also practically deployable in live cloud environments. This validation confirms that the predictive model generalizes well under actual workload conditions and provides reliable scaling decisions in practice.

4.1 Prediction Performance

The LSTM models demonstrated reliable forecasting capabilities for both EC2 and RDS instances. The predicted values for CPU utilization, network traffic, and I/O operations closely matched the actual observed values. For example, the model accurately forecasted a spike in CPU usage on an EC2 instance, prompting a successful scale-up before the resource hit its threshold. The mean absolute error (MAE) remained low across the test cases, indicating effective model generalization.

4.2 Scaling Decisions and Execution

Based on the predictions, the system performed dynamic scaling actions:

RDS Scaling: When CPU usage and ReadIOPS exceeded set thresholds, the system upgraded the instance type automatically. Similarly, when activity dropped significantly, a downgrade was initiated.

EC2 Scaling: The system initiated stop-modify-start sequences to adjust instance types. For example, low utilization led to a downgrade from t3.medium to t3.small, while high usage triggered upgrades.

In both cases, the decisions were executed using AWS Boto3 SDK with appropriate wait conditions, ensuring smooth transitions and minimal downtime.

4.3 Cost Optimization

One of the key benefits observed was cost reduction. By right-sizing underutilized instances and avoiding unnecessary overprovisioning, the system significantly lowered monthly billing compared to static instance allocation. A comparison chart showed a measurable decrease in resource charges without compromising performance.

4.4 Comparison with Traditional Methods

Traditional threshold-based auto-scaling reacts after usage exceeds predefined limits, often too late to prevent bottlenecks. In contrast, the developed system predicted future states, allowing scaling actions to occur before resource strain. This proactive approach improved performance consistency and user experience, especially under variable workloads.

4.5 System Robustness and Limitations

The real-time feedback loop and hourly decision cycle made the system adaptive and responsive. However, brief downtime was observed during RDS instance modification, which is a known constraint due to AWS infrastructure behavior. Future improvements could involve scheduling scaling actions during off-peak hours or using read replicas to maintain availability.

Overall, the results validate that the proposed AI-driven auto-scaling system enhances cloud resource efficiency, reduces operational costs, and maintains performance through proactive, intelligent scaling. The multi-metric forecasting model proved superior to traditional single-metric threshold systems in both accuracy and adaptability.

4.6 Advantages and Disadvantages of the Method

The developed AI-driven auto-scaling method offers several distinct advantages. By incorporating predictive modeling, it enables proactive resource allocation, which reduces the risk of system downtime and performance bottlenecks. Unlike traditional single-metric approaches, the multi-metric design provides a more comprehensive assessment of system load, capturing variations in CPU utilization, memory usage, and I/O activity. Moreover, the validation of the framework using real AWS data ensures its practical applicability and reliability in real-world cloud environments.

However, the system is not without limitations. Scaling operations in RDS instances involve brief downtime during modification, which may affect availability in latency-sensitive applications. Additionally, the inclusion of LSTM-based prediction introduces extra computational overhead for model training and inference, which may increase operational complexity. Finally, as the framework is tightly integrated with the AWS ecosystem, its portability to other cloud service providers is limited without further adaptation.

5. CONCLUSIONS

This work explored an AI-driven approach to cloud resource management by implementing LSTM models for predictive autoscaling of AWS EC2 and RDS instances. The developed system utilized historical data metrics from CloudWatch to predict resource demands and then enabling proactive scaling decisions. This method demonstrated improved prediction accuracy for major performance metrics, like CPU utilization, memory usage, and I/O operations. In the result, the system achieved a measurable reduction in cloud infrastructure costs and enhancing system responsiveness and scalability by having AI driven intelligent autoscaling for cloud resource optimization in AWS cloud framework

6. SUGGESTIONS AND RECOMMENDATIONS

Future research may integrate reinforcement learning for continuous adaptation and support multi-cloud environments. More granular metrics and multi-step forecasting could enhance robustness.

7. ACKNOWLEDGEMENTS

I would like to thank Asst. Prof. Daya Sagar Baral for supervision and my family for their support. The Department of Electronics and Computer Engineering , Pulchowk Campus, Institute of Engineering provided the necessary infrastructure and academic environment.

REFERENCES

[1] Rahane, Pushkar Deodatta, *Enhancing Cloud Efficiency Using Intelligent Autoscaling Algorithms*, Master's thesis, Dublin: National College of Ireland, 2023.

[2] Alharthi, Saleha, Alshamsi, Afra, Alseiari, Anoud, and Alwarafy, Abdulmalik, "Auto-scaling Techniques in Cloud Computing: Issues and Research Directions," *Sensors*, Vol. 24, No. 17, 2024.

[3] Netto, Marco A.S., Cardonha, Carlos, Cunha, Renato L.F., and Assuncao, Marcos D., "Evaluating Auto-scaling Strategies for Cloud Computing Environments," *Proceedings of the 2014 IEEE 22nd International Symposium on Modelling, Analysis and Simulation of Computer and Telecommunication Systems*, pp. 187–196, 2014.

[4] Gohel, Rupal and Jain, Gaytri, "Survey of Auto-scaling in Cloud Computing," *International Journal for Research and Development in Technology*, Vol. 4, No. 6, pp. –, December 2015.

[5] Pooja, Research Scholar, Jha, Kumari, and Pathak, Deepika, "Auto Scaling Techniques in Cloud Computing, 2024.

[6] Abdel Khaleq, Abeer and Ra, Ilkyeun, "Intelligent Autoscaling of Microservices in the Cloud for Real-Time Applications," *IEEE Access*, Vol. 9, pp. 35464–35476, 2021.

[7] Konidena, Shankar Dheeraj, "Impact of Autoscaling on Application Performance in Cloud Environments," *International Journal of Innovative Science and Research Technology*, pp. 1–5, October 2024.

[8] Pintye, István, Kovács, József, and Lovas, Róbert, "Enhancing Machine Learning-Based Autoscaling for Cloud Resource Orchestration," *Journal of Grid Computing*, Vol. 22, No. 4, December 2024.

[9] Gawande, Sachin and Gorthi, Anupam, "Containerization and Kubernetes: Scalable and Efficient Cloud-Native Applications," *International Journal of Innovative Science and Research Technology*, Vol. 9, pp. 435–439, November 2024.

[10] Christofidi, Georgia, Papaioannou, Konstantinos, and Doudali, Thaleia Dimitra, "Toward Pattern-Based Model Selection for Cloud Resource Forecasting," *Proceedings of the 3rd Workshop on Machine Learning and Systems (EuroMLSys '23)*, pp. 115–122, New York, NY: Association for Computing Machinery, 2023.