Predictive autoscaling in AWS Serverless by means of machine learning and SQS metrics*

Oleksandr Kyrychenko[1,*,†], Sergey Ostapov[1,†] and Oksana L. Kyrychenko[1,†]

[1] *Yuriy Fedkovych Chernivtsi National University, Kotsiubynskoho 2, 58012, Chernivtsi, Ukraine*

Abstract

Developing serverless computing platforms requires new approaches to scale computing resources optimally. Existing scaling mechanisms often cause performance issues, including cold starts, throttling, and increased costs due to the over-allocation of resources. This paper proposes an approach based on ML models that use Amazon SQS queue metrics to predict load and pre-scale Lambda functions in a distributed computing system. The research aims to develop a ready-to-use solution within the serverless computing model. The model estimates the queue depth and message arrival rate using time series forecasting using the DeepAR algorithm from AWS SageMaker. The MAE (mean absolute error) and RMSE (root mean square error) metrics assess the model's accuracy. The DeepAR model shows an MAE of 10019.21 and RMSE of 13140.91, indicating good prediction quality. The proposed solution demonstrates the capabilities of serverless platforms to solve the problem of predictive autoscaling being an effective tool for serverless applications. Empirical verification demonstrates improved performance indicators, including a decrease in the number of cold starts by (approximately) 27% and the number of unprocessed requests by (approximately) 14%.

Keywords

serverless, predictive modelling , predictive autoscaling , machine learning , neural networks, cloud

1. Introduction

The emergence and development of serverless computing platforms have changed the general approach to deploy and scale applications in cloud environments. With this approach, managing computing resources is mainly the cloud provider's responsibility, this allows developers to focus on writing code.

One key benefit of serverless platforms is the automatic scaling of applications in accordance with changes in load. Autoscaling implementation on such platforms requires that services should dynamically scale their resources, considering the change in the number of requests per unit of time. Thus, computing resources for user applications can be released entirely without traffic and reallocated when new requests appear, ensuring high resource efficiency.

The disadvantage of such active resource allocation is the notable initialization time. This disadvantage is known as the “cold start” problem when time is spent on deploying and configuring the runtime environment [1].

This problem becomes especially critical for services with short processing times for incoming requests, which are the majority in serverless environments, and is complicated by the fact that the created infrastructure is shared by several applications, each of which requires an individual approach to scaling solutions.

Intelitsis'25: The 6th International Workshop on Intelligent Information Technologies & Systems of Information Security, April 04, 2025, Khmelnytskyi, Ukraine

* Corresponding author.

† These authors contributed equally.

ol.kyrychenko@chnu.edu.ua (O. Kyrychenko); s.ostapov@chnu.edu.ua (S. Ostapov); o.kyrychenko@chnu.edu.ua (O. L. Kyrychenko)

0009-0001-6982-3342 (O. Kyrychenko); 0000-0002-4139-4152 (S. Ostapov); 0000-0003-0282-9958 (O. L. Kyrychenko)

One approach to address this problem is to apply machine learning techniques to predict workloads and automatically scale compute resources in AWS serverless environments [2]. This study proposes a framework that uses AWS SQS queue metrics to predict workloads in distributed and event-driven computing to pre-scale AWS Lambda functions. Applying time series prediction techniques based on the DeepAR algorithm in SageMaker allows us to minimize latency by avoiding "cold starts", throttling errors, and optimizing cloud computing costs [3].

The research goal is to develop a ready-to-use solution that allows us to maintain optimal performance with balanced resource utilization in cloud environments while remaining within the serverless computing model.

2. Related works

Many researchers are actively studying methods how to automatically scale computing resources in cloud environments. The goal of such studies is to increase application performance and ensure efficient use of cloud provider resources, which is especially critical for serverless architectures.

The traditional method for automatic scaling relies on specific resource usage or requests per second thresholds that trigger the scaling process. This approach is easy to implement but does not solve the problem of load spikes. For example, a cloud provider can maintain a number of prepared containers and delay deleting function instances after launches to efficiently handle new requests. However, this is economically inefficient since the user of serverless solutions pays only for the resources used regardless of when these resources were allocated. "Cold start" delays will still be present in the case of load spikes. The disadvantages and limitations of this approach are discussed in [4].

One approach considered in studies on automatic scaling for serverless applications is to reduce the number of "cold starts". The central aspect of such an approach is the ability to predict the load level to create the necessary resources proactively [5]. Typically, resource utilization metrics are used for forecasting, which allows determining the degree of necessary scaling. Although such approaches provide a quick response to changes in the load, they rarely consider the number of available resources at some point.

This becomes especially critical for serverless applications that use queues and event-driven architectures due to sudden changes in load and data sparsity [6]. Queues often store events during their processing, leading to additional delays and making it difficult to predict the load [7]. To address this issue, reference [8] combines queuing theory with methods for setting threshold values to manage virtual machine activation and deactivation. This allows service providers to optimize their financial costs but does not solve the problem of "cold start".

Solving the problem of "cold start" delays requires methods that allow predicting future workloads based on historical data and real-time indicators. Thus, in [9], the authors, using machine learning methods, were able to obtain a forecast that was very close to the actual data. The correct choice of the threshold value for microservices using the intelligent autoscaling system proposed in [10] improved the system's response time compared to the default autoscaling system offered by the cloud service provider.

Time series analysis is another popular technique for workload prediction. This technique examines historical data and uses statistical approaches to identify trends [4]. Machine learning and time series methods can create models to forecast message arrival rates in a distributed system, considering seasonality and event patterns [11].

A major challenge remains deploying machine learning models in the cloud and integrating them into existing serverless applications. The study [12] proposes a template for automated deployment of AWS services for training and evaluating models, which confirms the possibility of using predictive autoscaling in various serverless applications.

Despite many studies on automatic scaling and load forecasting of cloud resources, relatively little attention is paid to serverless computing models. Our research aims to create a solution for a

serverless distributed application that uses message queues to automatically scale resources based on metrics and machine learning algorithms.

3. Proposed methodology

We developed and launched a distributed application designed for batch data processing in AWS, which is used for load forecasting and automatic resource scaling. The application's main feature is its use of serverless technologies (Fig. 1).

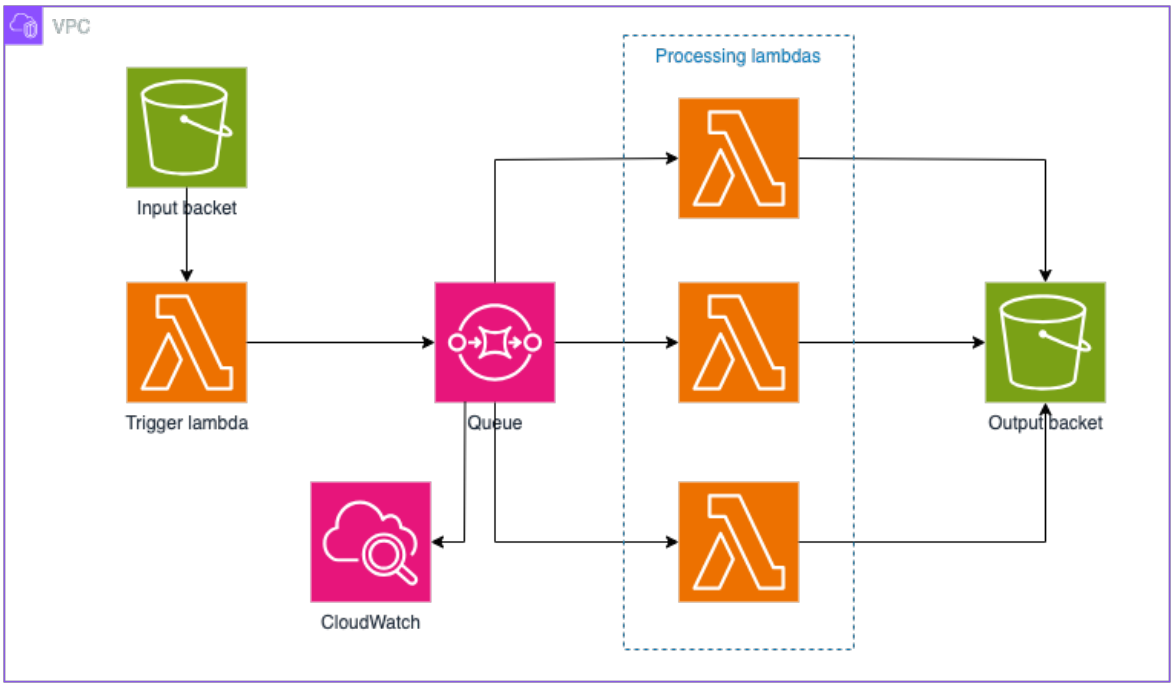

Figure 1: Serverless Application Architecture for Batch Data Processing Using AWS SQS.

The main task of the developed application is to ensure distributed processing of large amounts of data with a minimum number of errors. In these systems, a queue, typically AWS SQS, is the main component, receiving events and distributing data to a set maximum number of processors, like AWS Lambda. Depending on the current load, only the number of processors required at a particular moment in time within the quota is started. Input data and processing results are stored on S3.

Another component of our application is AWS CloudWatch [13], which monitors AWS SQS, namely resource usage metrics. From the perspective of the autoscaling problem, we will be interested in two metrics:

- ApproximateNumberOfMessagesVisible – queue length.
- NumberOfMessagesSent – number of messages sent [14].

To automatically scale the developed application, we proposed a framework based on AWS SageMaker [15]. This serverless service allows us to quickly build and deploy machine learning models with minimal developer intervention.

The proposed framework consists of several components for interaction with AWS SQS (Fig. 2). First of all, AWS CloudWatch collects the necessary metrics from AWS SQS, AWS EventBridge [16], and AWS Lambda [17]. AWS EventBridge launches AWS Lambda at appropriate intervals according to the schedule, and AWS Lambda receives predicted values from the trained model and adjusts the number and main parameters of data processors, thus ensuring horizontal scaling.

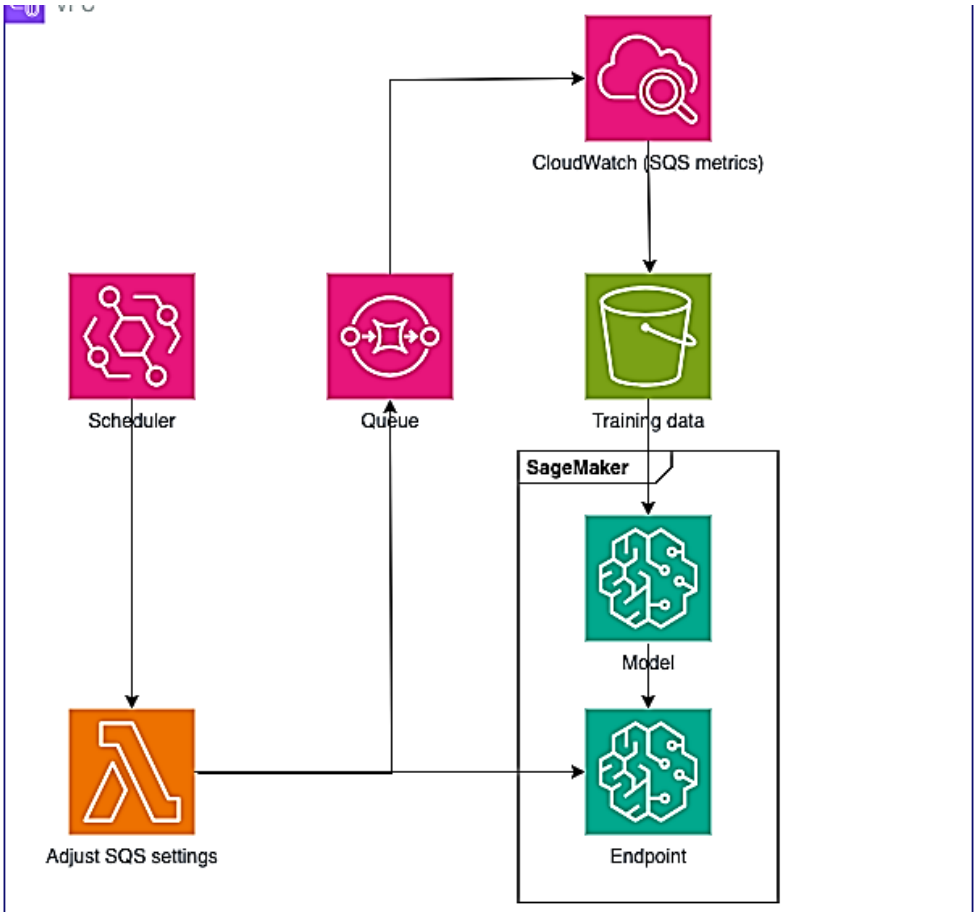

Figure 2: Framework architecture for automatically scaling data processors.

The diagram of framework and distributed application interaction is presented in Figure3.

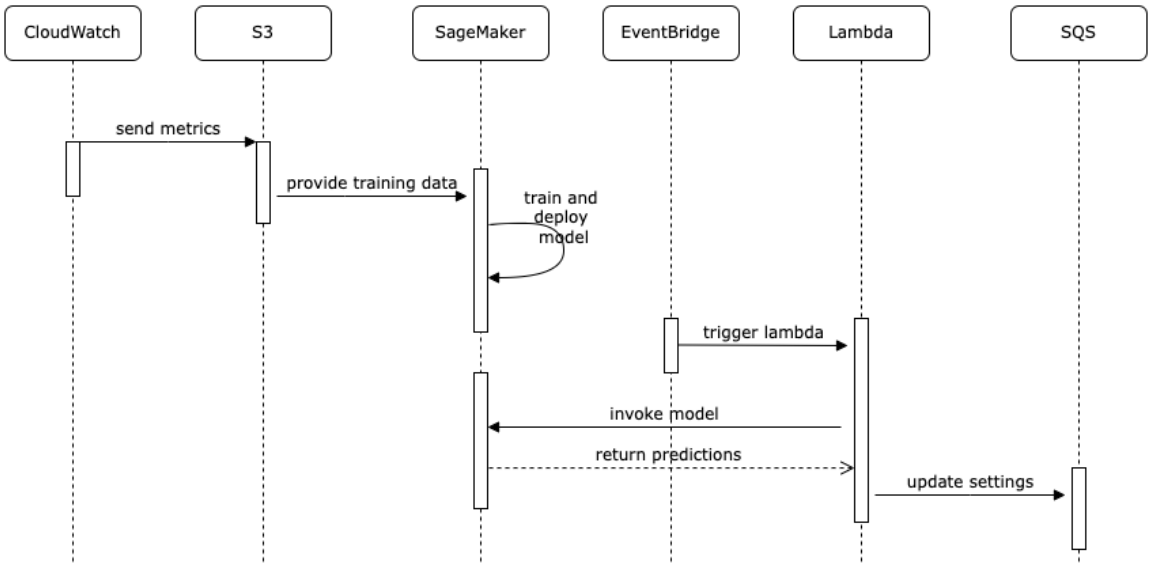

Figure 3: Framework and distributed application interaction diagram.

The interaction between the framework and the distributed application is as follows (Fig. 3):

1. AWS CloudWatch collects the necessary metrics from AWS SQS and stores them in S3.
2. A machine learning model is created and deployed in AWS SageMaker using stored metrics.
3. AWS EventBridge launches AWS Lambda according to a predefined schedule.

4. AWS Lambda accesses the deployed model and obtains predicted load indicators.
5. Based on the obtained indicators, AWS Lambda changes the settings of the data processors accordingly.

The model is built using DeepAR, a learning algorithm for time series forecasting using recurrent neural networks (RNNs). This algorithm can provide better accuracy than classical forecasting methods such as autoregressive integrated moving averages (ARIMA) or exponential smoothing (ES) [18, 19].

DeepAR boosts forecast accuracy by examining patterns from multiple related time series, unlike traditional methods that rely on a single series. DeepAR produces point and probabilistic forecasts, which is especially critical for capacity planning applications [19].

The accuracy of the forecast distribution is estimated using weighted quantile losses. The quantile loss for the quantile q∈(0,1) is calculated as follows (1):

$$QL(q, y, \hat{y}) = \begin{cases} q \cdot (y - \hat{y}), & if \ \ y > \hat{y} \\ (1 - q) \cdot (\hat{y} - y), & if \ \ y \leq \hat{y} \end{cases} \quad (1)$$

where:

q – the selected quantile (for example, 0.1, 0.5, or 0.9),

y – actual value,

$\hat{y}$ – forecast for the corresponding quantile.

For a set of quantiles $Q = \{q_1, \ q_2, \dots, \ q_k\}$ the total weighted quantile loss is calculated as (2):

$$QL_{weighted} = \frac{1}{n} \sum_{i=1}^{n} \sum_{q \in Q} \omega_q \cdot QL\left(q, y_i, \hat{y}_i^{(q)}\right) \quad (2)$$

where [20]:

n – number of predicted points;

ω_q – weight for quantile q;

$\hat{y}_i^{(q)}$ – the prediction for the quantile$q$ for the point i.

Also used [21, 22] were the mean absolute error (3) and the root mean square error (4):

$$MAE = \frac{1}{n} \sum_{i=1}^{n} |y_i - \hat{y}_i| \, , \quad (3)$$

$$RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2}. \quad (4)$$

Building a DeepAR model consists of the following steps:

1. Collecting and formatting AWS SQS metrics.
2. Defining and training hyperparameters.
3. Training the model.
4. Testing the model.
5. Evaluating the results.
6. Deploying the model for further use.

The obtained prediction results are used to automatically scale data processors according to the algorithm below using threshold values.

Algorithm 1. AWS Lambda provisioned concurrency changing

```
function AutoScaleLambda(f_name, t_up, t_down, min_c, max_c, inc, target)
  // Main loop that runs every N minutes
   loop every N minutes
      visible ← GetPrediction(target)
      current_c ← GetProvisionedConcurrency(f_name)
      // If predicted value exceeds upper threshold
      if visible > t_up then
         new_c ← current_c + inc
         if new_c > max_c then
            new_c ← max_c
         end if
         // Update provisioned concurrency
         SetProvisionedConcurrency(f_name, new_c)
      else if visible < t_down then
         new_c ← current_c - inc
         if new_c < min_c then
            new_c ← min_c
         end if
        // Update provisioned concurrency
         SetProvisionedConcurrency(f_name, new_c)
      end if
     // Wait for N minutes before next iteration
      Sleep(N minutes)
   // end loop
   end loop
end function
```

where:
f_name = Lambda function name,
t_up = Scale-up threshold for ApproximateNumberOfMessagesVisible,
t_down = Scale-down threshold for ApproximateNumberOfMessagesVisible,
min_c = Minimum provisioned concurrency,
max_c = Maximum provisioned concurrency,
inc = Increment step for scaling,
target = Historical time series values.

The framework built in this way will automatically scale data processors in distributed applications using queues.

4. Results and discussion

Our experimental setup consists of a distributed application and a component for predicting and automatically scaling data processors in the distributed application. The core of the application is a queue – a component designed to store and send events for processing. Each event contains the information necessary for successful processing.

To simulate the workload, Locust was used. This tool is effective for load-testing distributed systems due to its scalability, ease of scripting, and monitoring and analysis capabilities [23].

During the application testing, the queue received 1082178 events within 90 minutes. At the same time, we collected the approximate number of messages visible and number of messages sent metrics for AWS SQS. Figure 4 and Figure 5 illustrate the collected data distribution.

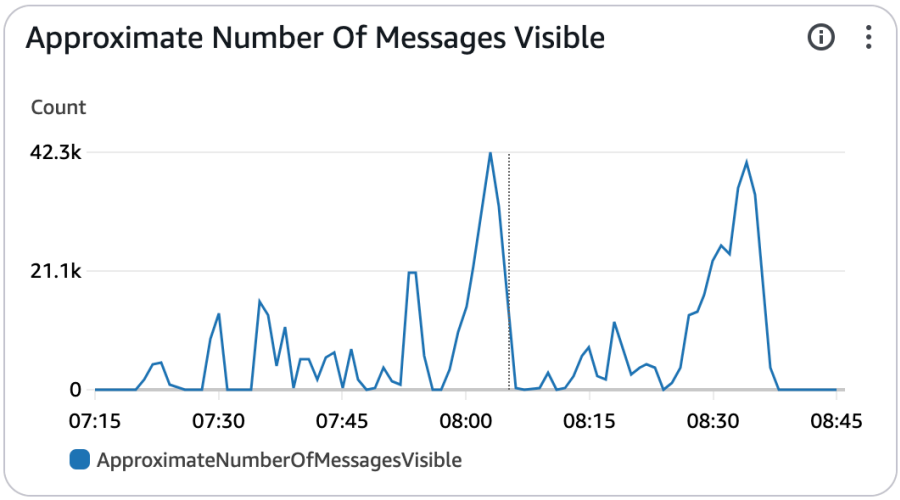

Figure 4: Approximate number of messages visible.

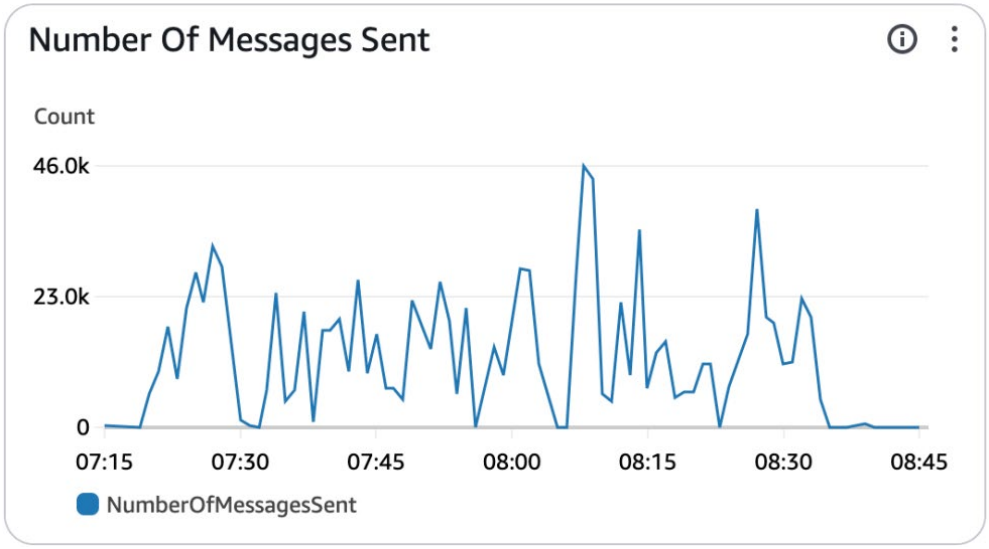

Figure 5: Number of messages sent.

The collected data was used to train the DeepAR model. It is worth noting that DeepAR uses a recurrent LSTM neural network (RNN). For training, we chose a minute-by-minute time series granularity. The model required at least 30 steps of historical data for forecasting, with an initial forecast length of 50 minutes.

The training results are presented in Figure 6 and Figure 7.

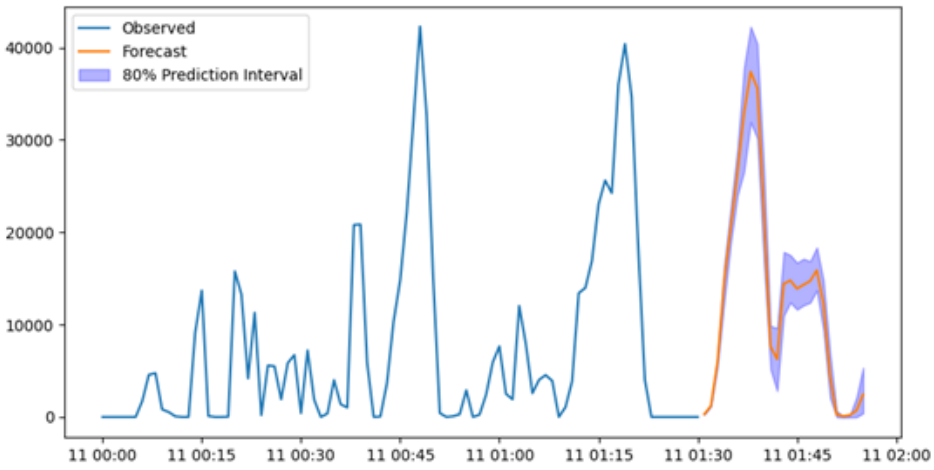

Figure 6: Approximate number of messages visible predicted values.

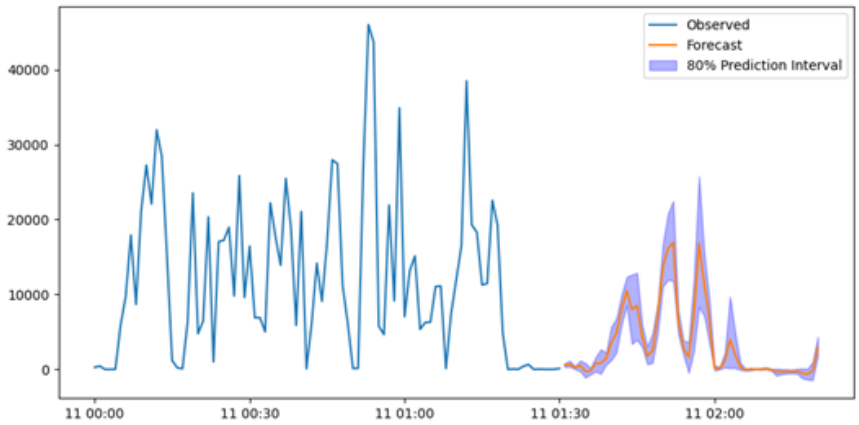

Figure 7: Number of messages sent predicted values.

To evaluate the resulting models, we used the mean absolute error and the root mean square error for different numbers of layers. Figure 8 and Figure 9 represent a differentiation of the obtained error values.

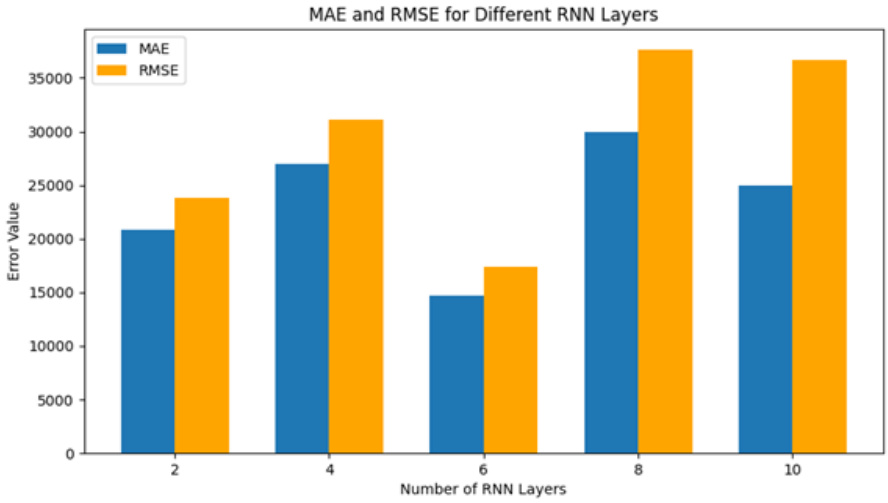

Figure 8: MAE and RMSE for different number of RNN layers for Approximate number of messages visible predicted values.

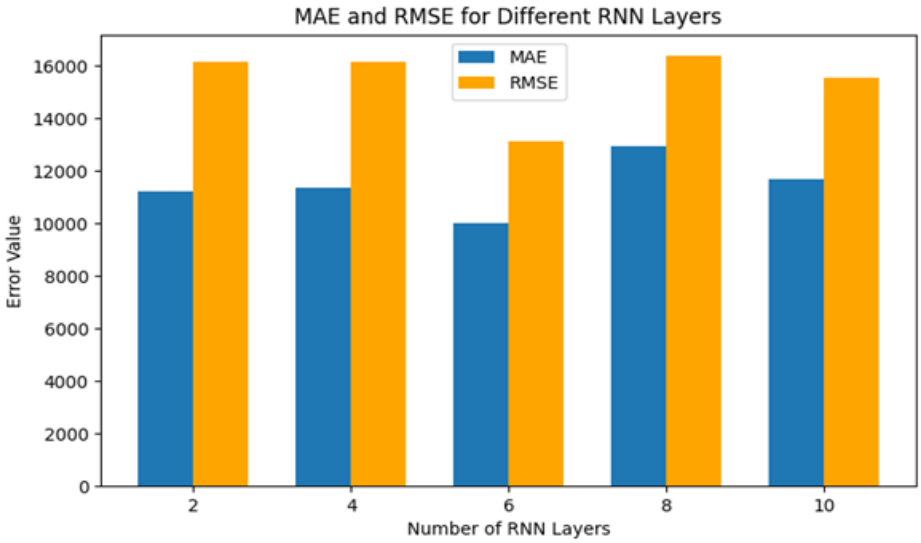

Figure 9: MAE and RMSE for different number of RNN layers for Number of messages sent predicted values.

By varying the number of layers, we found that the models demonstrate the best performance at a value of 6 and can be used to predict the load of a distributed application.

The trained models were deployed in the cloud environment using AWS SageMaker, which allowed us to make queries to predict the values of selected AWS SQS metrics and reduce the number of cold starts (Fig. 10).

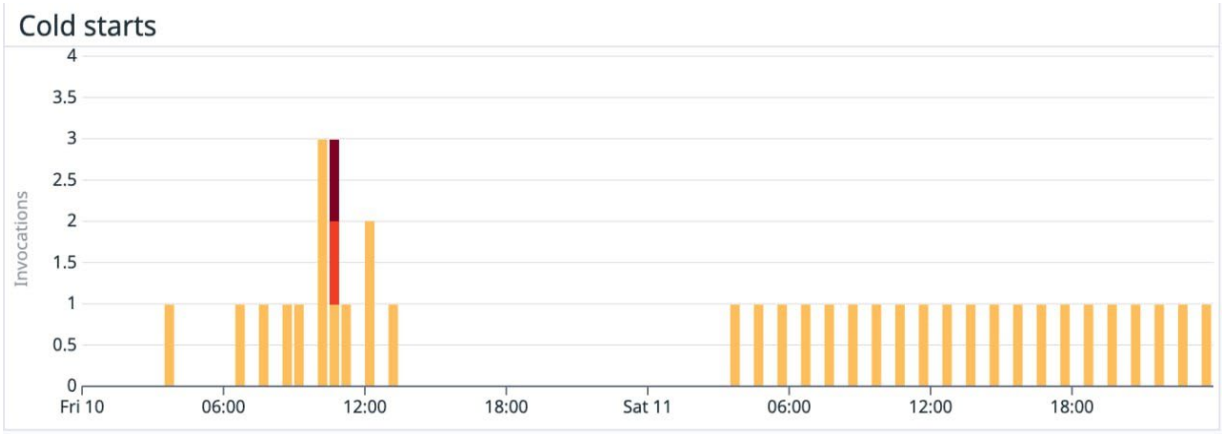

Figure 10: The number of cold starts after applying predictive autoscaling.

5. Conclusion

Our study presents a framework for predictive automatic scaling of computing resources in serverless environments. The framework enables a transition from reactive to proactive scaling, reducing “cold starts” and overload errors.

The DeepAR model shows a mean absolute error (MAE) of 10019.21 and a root mean square error (RMSE) of 13140.91, indicating good prediction quality. Combining the proposed framework with serverless applications contributes to the optimal use of computing resources without going beyond the concept of serverless computing.

It should be noted that using prediction to scale data processors automatically reduced the number of “cold starts” by (approximately) 27% and the number of unprocessed requests by (approximately) 14%.

One of the limitations of the proposed framework is the lack of historical data, which DeepAR needs to learn patterns. In scenarios where a time series has very few past observations (e.g., newly created services), DeepAR may fail to model underlying trends, reducing forecast reliability accurately [24].

Further research can focus on improving the quality and use of forecasting models. For example, DeepAR allows building models based on multiple time series, which helps improve forecast accuracy. A separate issue remains in determining the optimal values of hyperparameters for training the DeepAR model. In addition, data sparsity and model drift require constant automatic retraining.

Using additional SQS metrics enables essential horizontal and vertical scaling for queues with varying event processing times.

Another important direction is the study of more efficient implementation of computing infrastructure costs through automatic scaling [25]. Also, we can create the SageMaker pipeline to update the DeepAR model with the latest data automatically.

The proposed framework is a practical automatic scaling tool for serverless applications that improves performance and reliability in distributed systems.

Declaration on Generative AI

During the preparation of this work, the authors used ChatGPT, Grammarly in order to: Grammar and spelling check, Paraphrase and reword. After using this tool/service, the authors reviewed and edited the content as needed and take full responsibility for the publication's content.

References

[1] S. Thumala, Building Highly Resilient Architectures in the Cloud, Nanotechnology Perceptions 16 (2020) 164-284.

[2] P. Gupta, M.K. Goyal, S. Chakraborty, A.A. Elngar, Machine Learning and Optimization Models for Optimization in Cloud, 1st ed., Chapman and Hall/CRC, 2022. doi:10.1201/9781003185376.

[3] A. Mampage, Autonomous Resource Management for Serverless Computing, Ph.D. thesis, School of Computing and Information Systems the University of Melbourne, Melbourne, Australia, 2023.

[4] S. Alharthi, A. Alshamsi, A. Alseiari, A. Alwarafy, Auto-Scaling Techniques in Cloud Computing: Issues and Research Directions, Sensors 24 (2024) 5551. doi:10.3390/s24175551.

[5] A. Singhvi, A. Balasubramanian, K. Houck, M.D. Shaikh, S. Venkataraman, A. Akella, Atoll: A scalable low-latency serverless platform, in: Proceedings of the ACM Symposium on Cloud Computing, SoCC '21, ACM Press, New York, NY, USA, 2021, pp. 138–152. doi:10.1145/3472883.3486981

[6] E. Jonas, J. Schleier-Smith, V. Sreekanti, C.-C. Tsai, A. Khandelwal, Q. Pu, V. Shankar, P. Sarda, Cloud Programming Simplified: A Berkeley View on Serverless Computing, arXiv preprint arXiv:1902.03383, 2019. doi:10.48550/arXiv.1902.03383.

[7] L. Wang, M. Zhang, J. Han, G. Chen, H. Chen, Peeking Behind the Curtains of Serverless Platforms, in: Proceedings of the 2018 USENIX Annual Technical Conference (ATC '18), USENIX Association, Boston, MA, 2018, pp. 133–146.

[8] T. Tournaire, H. Castel-Taleb, E. Hyon, Efficient Computation of Optimal Thresholds in Cloud Auto-scaling Systems, in: ACM Transactions on Modeling and Performance Evaluation of Computing Systems, volume 8, issue 4, article no. 9, ACM, 2023, pp. 1–31. doi:10.1145/3603532.

[9] M.B. Taha, Y. Sanjalawe, A. Al-Daraiseh, S. Fraihat and S.R. Al-E'mari, Proactive Auto-Scaling for Service Function Chains in Cloud Computing Based on Deep Learning, in: IEEE Access, volume 12, 2024, pp. 38575-38593. doi: 10.1109/ACCESS.2024.3375772

[10] A.A. Khaleq, I. Ra, Intelligent Autoscaling of Microservices in the Cloud for Real-Time Applications, in: IEEE Access, volume 9, 2021, pp. 35464-35476. doi:10.1109/ACCESS.2021.3061890.
[11] H. Muccini, K. Vaidhyanathan, A Machine Learning-Driven Approach for Proactive Decision Making in Adaptive Architectures, in: IEEE International Conference on Software Architecture Companion (ICSA-C), Hamburg, Germany, 2019, pp. 242-245, doi:10.1109/ICSA-C.2019.00050.
[12] A. M. Nestorov, Optimizing Serverless Architectures for Data-Intensive Analytics Workloads, Ph.D. thesis, Universitat Politècnica de Catalunya - BarcelonaTECH, 2024.
[13] What is Amazon CloudWatch? 2025. URL: https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/WhatIsCloudWath.html
[14] Amazon Simple Queue Service, 2024. URL: https://aws.amazon.com/sqs
[15] Amazon SageMaker, 2024. URL: https://aws.amazon.com/sagemaker/
[16] Amazon EventBridge, 2024. URL: https://aws.amazon.com/eventbridge/
[17] What is AWS Lambda? 2025. URL: https://docs.aws.amazon.com/lambda/latest/dg/welcome.html.
[18] Use the SageMaker AI DeepAR forecasting algorithm, 2025. URL: https://docs.aws.amazon.com/sagemaker/latest/dg/deepar.html.
[19] T. Januschowski, D. Arpin, D. Salinas, V. Flunkert, J. Gasthaus, L. Stella, P. Vazquez, Now available in Amazon SageMaker: DeepAR algorithm for more accurate time series forecasting, 2018. URL: https://aws.amazon.com/blogs/machine-learning/now-available-in-amazon-sagemaker-deepar-algorithm-for-more-accurate-time-series-forecasting/
[20] D. Salinas, V. Flunkert, J. Gasthaus, T. Januschowski, DeepAR: Probabilistic forecasting with autoregressive recurrent networks, International Journal of Forecasting 36 (2020) 1181–1191. doi:10.1016/j.ijforecast.2019.07.001.
[21] A. Samad, I. M. Khan, Rahaman, M. A. Islam, Data-driven approach to predict future oil production of an Oil Field using machine learning techniques, in: Proceedings of the 8th International Conference on Mechanical, Industrial and Energy Engineering 2024, Khulna University of Engineering and Technology, 2025.
[22] D. Bliedy, M. H. Khafagy, R. M. Badry, Dynamic Resource Utilization Prediction Model for Cloud Datacenter, IAENG International Journal of Applied Mathematics 55 (2025) 307–331.
[23] J. Heyman, L. Holmberg, A. Baldwin, C. Byström, J. Hamrén, H. Heyman. What is Locust? 2025. URL: https://docs.locust.io/en/stable/what-is-locust.html.
[24] K. Yemets, Time Series Forecasting Model For Solving Cold Start Problem Via Temporal Fusion Transformer, Computer Systems and Information Technologies 1(2024) 57–64. doi: 10.31891/csit-2024-1-7.
[25] S. Ponnusamy, M. Khoje, Optimizing Cloud Costs with Machine Learning: Predictive Resource Scaling Strategies, in: 2024 5th International Conference on Innovative Trends in Information Technology (ICITIIT), Kottayam, India, 2024, pp. 1-8. doi: 10.1109/ICITIIT61487.2024.10580717.