This study's object is the organization of distributed data processing in cloud environments using serverless computing.

The evolution of serverless computing has led to a change in approaches to building the architecture of applications deployed in the cloud. The ability to abstract from infrastructure management while achieving cost-effectiveness is becoming a significant task requiring new tools. One such tool is a new framework that makes it possible to scale computing resources depending on the workload dynamically, store the state of the computing process using DynamoDB, and provide real-time progress tracking.

A serverless application for the distributed generation of PDF documents has been developed and deployed to test the proposed framework. The real load was emulated using Locust; files containing 1,025,132 records were fed to the application input. The results of the experiments showed that the application started to work 25.8% faster, the throughput increased by 21.3%, and the number of cold starts decreased by 3% compared to conventional scaling.

Additionally, the main areas of further research on developing and improving the designed framework have been identified. This study provides possibilities for predictive automatic scaling using semi-Markov process models in a serverless environment.

Unlike traditional reactive approaches, the proposed approach predicts changes in advance and proactively scales parts of the application, which makes it possible to reduce delays and avoid cold starts. The framework could be used to develop serverless applications for distributed data processing using message queues and the ability to monitor the processing in real time

Keywords: serverless architecture, predictive autoscaling, cloud computing, distributed data processing

UDC 004.41, 004.75
DOI: 10.15587/1729-4061.2025.335723

DESIGN OF A FRAMEWORK FOR SERVERLESS DISTRIBUTED DATA PROCESSING USING QUEUES

Oleksandr Kyrychenko
Assistant*
Serhii Ostapov
Corresponding author
Doctor of Physico-Mathematical Sciences, Professor
Department of Computer Systems Software**
E-mail: s.ostapov@chnu.edu.ua
Oksana Kyrychenko
PhD, Associate Professor*
*Department of Mathematical Problems of Control and Cybernetics**
**Yuriy Fedkovych Chernivtsi National University
Kotsiubynskoho str., 2, Chernivtsi, Ukraine, 58002

Received 13.05.2025
Received in revised form 02.07.2025
Accepted date 17.07.2025
Published date 29.08.2025

How to Cite: *Kyrychenko, O., Ostapov, S., Kyrychenko, O. (2025). Design of a framework for serverless distributed data processing using queues. Eastern-European Journal of Enterprise Technologies, 4 (9 (136)), 19–25. https://doi.org/10.15587/1729-4061.2025.335723*

1. Introduction

Given the growing popularity of cloud services, serverless computing has become one of the best options when choosing an architecture for developing cloud applications. The ability to get rid of the problems associated with infrastructure management, automatically scaling the required resources, and achieving the necessary cost efficiency has made serverless architectures especially attractive for developing applications with distributed computing and high workloads. This approach makes it possible to decompose a monolithic application into a number of independent cloud functions. Developers can focus on implementing business logic, leaving everything else to cloud service providers. That is why serverless computing is widely used in the development of event-driven applications and applications where there is no need to store state directly in the application itself.

Despite significant advantages, serverless computing has a number of limitations, which significantly complicates the process of data processing in distributed environments. First of all, this is the limited execution time of the cloud function, which requires splitting the main task into independent subtasks of the appropriate volume and coordinating data processing in subtasks. Another constraint is the dependence of any serverless application on reactive scaling mechanisms when additional resource allocation occurs in response to a change in load [1]. This, in turn, leads to an increase in the number of cold starts and problems with response delay. In addition, the lack of state in serverless applications makes it difficult to track the progress of individual tasks in real time.

Quite often, distributed applications face the problem of establishing optimal interaction between individual components, which can be completely independent and implemented using different technologies and programming languages. For this purpose, developers use message queues that provide asynchronous communication between components and event buffering. However, this approach does not offer built-in mechanisms for predictive scaling, which leads to inefficient use of available resources and an increase in the overall execution time of the main task.

In this regard, scientific research aimed at designing tools for dynamically scaling computing resources depending on the workload remains extremely relevant. The results of such research are important for practice as they allow for the rapid deployment of distributed systems in the cloud environment that are able to respond effectively to changes in load.

2. Literature review and problem statement

In [2], adaptive methods for initial resource allocation are investigated to improve the use of the provider's infrastructure. The authors propose an algorithm for optimal scheduling of functions taking into account the nature of the loads. The proposed algorithm makes it possible to achieve a certain balance between execution delay and cost. However, mechanisms for predicting load changes in real time are not taken into account.

Studies [3, 4] emphasize the importance of prioritizing tasks in distributed systems to improve performance and reduce delays. The main task is to determine criteria and build models for predicting the load for message queue management, which is an important factor in the case of limited resources. However, adaptive resource changes in a serverless environment are not taken into account.

Another mechanism for accelerating data processing in distributed systems is batch processing. In [5], methods are proposed that make it possible to combine several tasks into batches and process them as a single unit. This approach helps reduce the number of cold starts of a function and increases the overall efficiency of the system, especially under high load conditions. In addition, special attention was paid to reducing data processing delays by using various queue management strategies. However, there is almost no research on how such approaches interact with autoscaling systems due to the limitations of modern FaaS platforms in controlling the execution of functions.

An important approach to solving the problems of autoscaling serverless applications is to utilize the capabilities of Amazon Web Services (AWS) Simple Queue Service (SQS) [6, 7]. In this case, the trigger for starting to scale is a change in key queue characteristics, such as the number of messages waiting to be processed or the number of messages that have already been processed. The authors paid attention to the impact of different SQS configurations on the overall behavior of the system and provided recommendations for selecting and configuring a queue for specific use cases. However, the issues of cold start, as well as load forecasting, were left out of consideration, primarily due to the lack of openness of cloud providers regarding their internal infrastructure and the lack of access to automatic ML platforms at the time of publication.

It is worth noting that properly configured autoscaling leads to a decrease in the number of cold starts in the application. Conventional methods for autoscaling resources are based on threshold values and generally demonstrate good results but experience certain difficulties in the cases of sudden load increases. Of course, a cloud provider can keep prepared, pre-warmed containers to smooth out peak loads, but this is not always economically feasible [8]. However, the issue of integrating serverless approaches with predictive scaling remains open. The reason is the difficulty of building universal models that can adapt to a wide range of applications.

A number of studies suggest the use of predictive autoscaling using machine learning models. Owing to this approach, it is possible to plan the use of computing resources in advance [9]. However, quite often, methods for predictive autoscaling do not take into account the actual availability of resources, especially in applications that use queues, which means that there is a need to monitor queue parameters to make a decision on scaling [10].

The ability to implement real-time communications using a serverless architecture has significant advantages and is an attractive option for modern web applications. AWS AppSync or WebSockets via API Gateway reduce latency in real-time message delivery and simplify the architecture for applications with high message rates. These services ensure that updates are delivered to users in real time, increasing the responsiveness and interactivity of applications. Latency is minimized by using globally distributed infrastructure and edge locations [11]. However, the cited paper does not analyze how real-time tools function in distributed computing.

Studies [12, 13] consider methods of connection managee ment in a serverless environment and propose approaches to reduce costs while maintaining high performance. In general, the use of serverless architecture for real-time communications provides a corresponding economic effect, provided that the correct implementation strategy is chosen and the load in RTC applications is irregular. However, the authors of the studies did not propose solutions for load forecasting or resource planning in serverless environments. The reason is the uncertainty of QoS parameters in an unpredictable environment.

Despite a significant body of research into the optimal use of resources in distributed computing in cloud environments, approaches specific to serverless architectures remain insufficiently studied. Most are limited to reactive methods or solve partial aspects. The peculiarities of the division of responsibility between the cloud provider and developers do not make it possible to fully control the application behavior and complicate the search for optimal configurations. The lack of complete information regarding resource management by a cloud provider makes it difficult to conduct research and design serverless solutions.

3. The aim and objectives of the study

The purpose of our research is to design a new structure – a framework for distributed data processing using a serverless architecture. Considering the reactivity of conventional approaches to autoscaling, which leads to delays during sudden load changes in distributed systems using queues, the framework offers an approach for load forecasting and proactive resource scaling.

To achieve this aim, the following objectives were accomplished:

– to design the architecture of the framework;

– to implement a predictive autoscaling module to ese timate the probability of the system transitioning between different load states and the corresponding change in the configuration of queues and data processors;

– to compare the designed framework with conventional autoscaling.

4. The study materials and methods

The object of our study is the implementation of distributed data processing in cloud environments using serverless technologies and message queue management systems.

The hypothesis of the study assumes that the use of serverless architectures in combination with message queues (in particular, AWS SQS) allows for effective automatic scaling of distributed data processing, reducing system response time, the number of cold starts, and increasing overall performance. This approach implements weak connectivity of individual system elements and provides scaling of both the system as a whole and its individual parts if necessary.

The following assumptions were accepted when conducting the simulation:

– all messages entering the queue are valid and have the same size;

– scaling of data processors is carried out within the quou ta provided by the cloud provider and does not depend on the internal limitations of the cloud platform;

– all system components are deployed within the same AWS region to minimize the impact of delays in network connections.

The research method is an experimental approach, which involves the development and testing of prototypes of a serverless application for distributed generation of PDF documents using predictive and conventional autoscaling and further comparison of the results. Files in csv format containing 1,025,132 records were used as input data. A single-page PDF document was generated based on each record. Records were divided into groups of 50 elements per group and sent to the queue with a delay of 1 second.

5. Research results: a framework for distributed data processing

5. 1. Framework architecture

The framework architecture uses the following AWS services:

– AWS S3 for storing input data and calculation results [14];
– Amazon SQS as a task queue [15];
– AWS Lambda for event-based calculations [16];
– DynamoDB for storing data processing state [17];
– AWS AppSync for monitoring data processing state in real time [18];
– Amazon SageMaker for deploying a load forecasting model [19];
– AWS EventBridge as a scheduler for launching AWS Lambda on a schedule [20].

The overall architecture of the framework is shown in Fig. 1 and is built on a multi-tiered principle, where each tier has a clearly defined area of responsibility. This approach makes it possible to provide the necessary isolation of individual components and accordingly improve the scalability of the system, simplify its development and maintenance, and facilitate testing. The basic levels of the framework are:

– data loading layer;
– data processing layer;
– state management layer;
– real-time monitoring layer;
– predictive autoscaling layer.

Data loading layer.

This layer uses a batch data processing mechanism and a message queue. Messages are processed asynchronously, which provides high throughput with minimal latency.

Data processing layer.

When new messages appear in the queue, AWS Lambda functions are launched. The main task of Lambda functions is to process data and save information about the processing state.

State management layer.

DynamoDB is used to save the data processing state. The presence of a saved state makes it possible to monitor the execution of the processing process and track errors.

Real-time monitoring layer.

To organize real-time communication with connected clients, the framework uses AWS AppSync. This approach makes it possible to receive information about the progress of distributed data processing with virtually no delays.

Predictive autoscaling layer.

An important element of the framework is the predictive autoscaling module, which receives information about the input data and estimates the probability of the system transitioning between different load states. Based on the forecasts, the main parameters of AWS Lambda and SQS are adjusted to optimize resource usage. Forecasting is performed every 5 seconds, while the model assumes state changes with a horizon of 30 seconds.

The framework operation scheme is shown in Fig. 2.

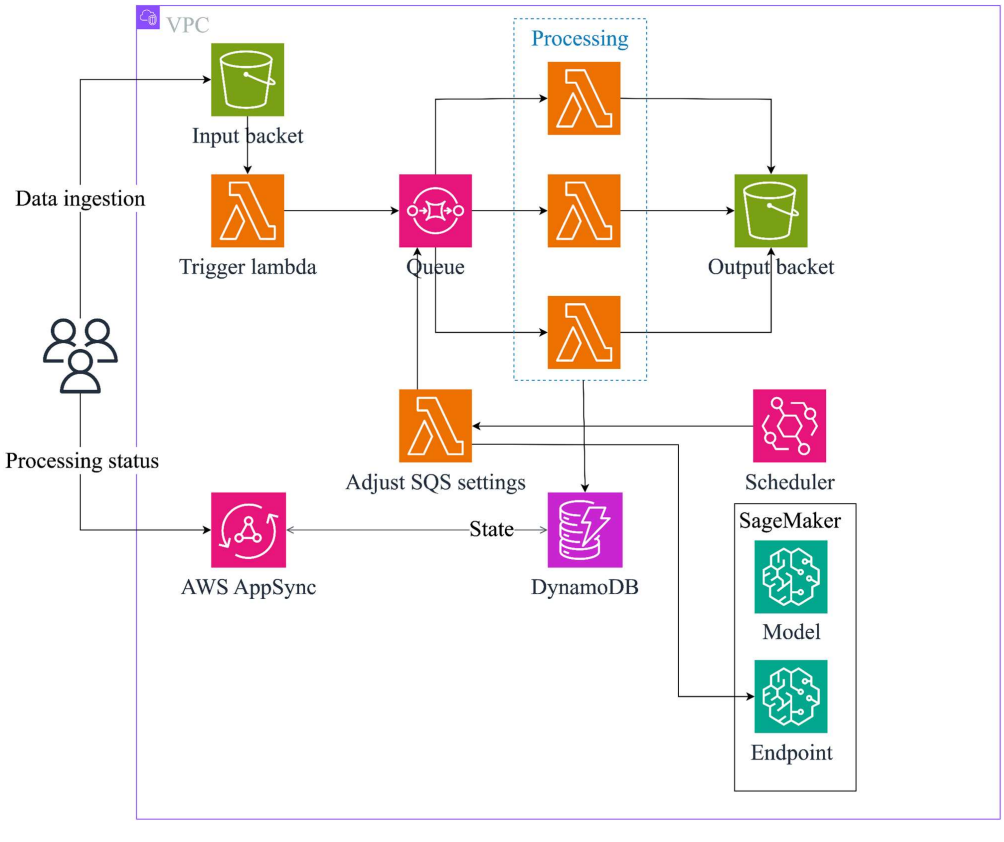

Fig. 1. Framework architecture for distributed data processing

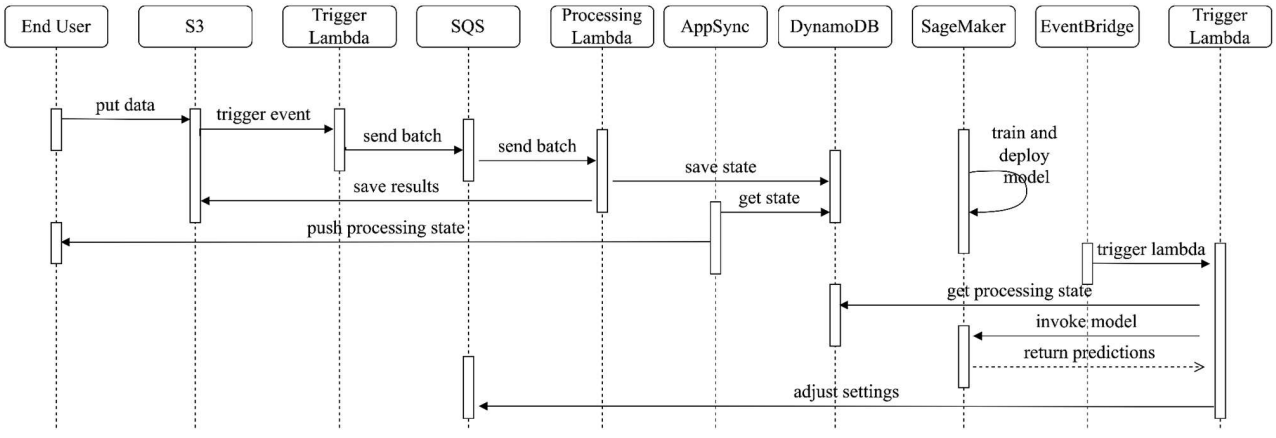

Fig. 2. Framework and interaction diagram of a distributed application

It covers the following main stages:

– User uploads data to S3 for processing;

– S3 activates Trigger Lambda;

– Trigger Lambda prepares task packages and sends mese sages to the queue;

– Queue launches one or more Processing Lambdas and passes the received messages to the input;

– Processing Lambda processes the data, stores the result in S3, and updates the processing state in DynamoDB;

– DynamoDB sends AppSync updates;

– AppSync broadcasts the processing state updates to the user;

– Scheduler runs a custom AWS Lambda function on schedule, which contacts SageMaker Endpoint for a forecast of future load and updates SQS and Processing Lambda configuration parameters.

5. 2. Predictive autoscaling module

The main functional purpose of the predictive autoscaling module is to determine the probability of changing the system operating modes under the influence of the load and adjust the main parameters of queues and data processors. For load prediction, semi-Markov models deployed in Amazon SageMaker are used, which are effective for modeling dynamic events, such as load changes in distributed serverless architectures. Compared to conventional Markov systems, they provide greater realism since they do not limit the duration of stay in states to only exponential distributions. This property is useful for applications with variable load, which contributes to more accurate prediction of resource usage and performance assessment. In addition, semi-Markov processes are effective for solutions with multiple operating modes [21]. Models are trained based on various load indicators, such as the number of incoming events, the average processing time of each event, the length of the queue, the number of AWS Lambda functions launched. SageMaker uses this data to construct transition probabilities and delay time distributions in each state.

The SQS and Processing Lambda configuration parameters are modified based on the prediction received from SageMaker according to the following algorithm:

Algorithm 1. AWS Lambda and SQS parameters adjusting

1: function PredictiveAutoScaler(SMC_Model, P_INTERVAL, THRESHOLDS)
2: loop every P_INTERVAL seconds
3: metrics ← CollectMetrics ()
4: sqs ← metrics.sqs_message_count
5: λ ← metrics.lambda_concurrency
6: current_state ← ClassifyState(sqs, THRESHOLDS)
7: prediction ← QuerySageMaker(SMC_Model, current_state, metrics)
8: next_state ← prediction.next_state
9: if next_state ∈ {High_Load, Overload} then
10: IncreaseLambdaConcurrency()
11: IncreaseSQSPollingRate()
12: else if next_state = Low_Load then
13: DecreaseLambdaConcurrency()
14: DecreaseSQSPollingRate()
15: end if
16: PushAppSyncUpdate(current_state, next_state, prediction.transition_time)
18: Sleep(P_INTERVAL seconds)
19: end loop
20: end function

where:

– SMC_Model – semi-Markov model;

– P_INTERVAL – time after which a new forecast will be obtained;

– THRESHOLDS – threshold values for determining the state.

Based on the forecast data, the main parameters of queues and handlers are adjusted to optimize the use of computing resources. The combination of semi-Markov models with the given algorithm in the form of a module makes it possible to predict spikes in workload and automatically scale computing resources.

5. 3. Comparison of the designed framework with classic autoscaling

To test the proposed framework, a serverless application for distributed generation of PDF documents was developed. The application was deployed on AWS in two versions with predictive autoscaling and without predictive autoscaling. Locust was used to emulate the workload, which is an effective tool for load testing of distributed systems. The application architecture fully uses the capabilities of the framework, where the main component is a message queue, which provides weak connectivity of other application components. This in turn makes it possible

to easily scale the necessary parts of the system, recover from failures faster, and use available resources more efficiently.

The developed prototype was fed with record files as input. Each record contained data for generating a single-page PDF file. In total, the downloaded files contained 1,025,132 records. Records were divided into groups of 50 elements per group (this value was obtained empirically in previous studies [22]) and sent to the queue with a delay of 1 second.

During testing, several of the most critical indicators were collected and analyzed, such as data delay time in processing, the number of messages processed per unit of time, the number of cold starts.

Fig. 3 shows the change in data processing time in the developed application over 5 minutes. It is worth noting that the use of predictive autoscaling accelerated data processing by an average of 25.8%.

The ability to dynamically adjust SQS and Processing Lambda configuration parameters resulted in an average 21.3% increase in system throughput.

The comparison chart in Fig. 4 demonstrates the change in application behavior using predictive autoscaling. As can be seen from Fig. 4, reactive scaling does not always lose to predictive scaling, that is, there are other factors whose influence does not guarantee the required result.

One such factor is the problem of cold starts. Although predictive autoscaling has reduced the proportion of cold starts to approximately 3% (Fig. 5), it has not been possible to completely eliminate the negative effect of this phenomenon.

Overall, predictive autoscaling stabilized the behavior of the serverless application in terms of resource usage, especially in the cases of sudden load changes, primarily due to the ability to make timely scaling decisions.

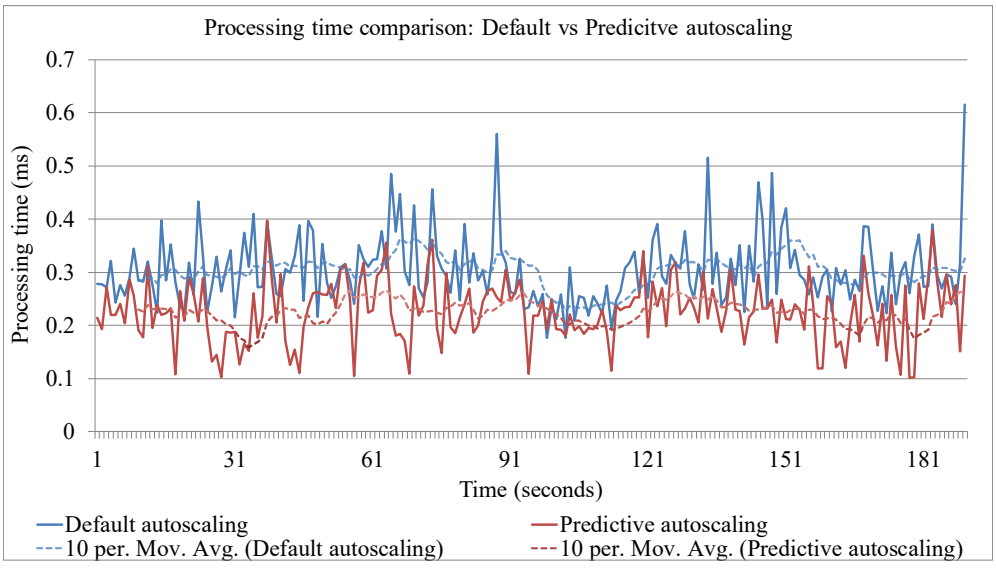

Fig. 3. Response time: default and predictive autoscaling

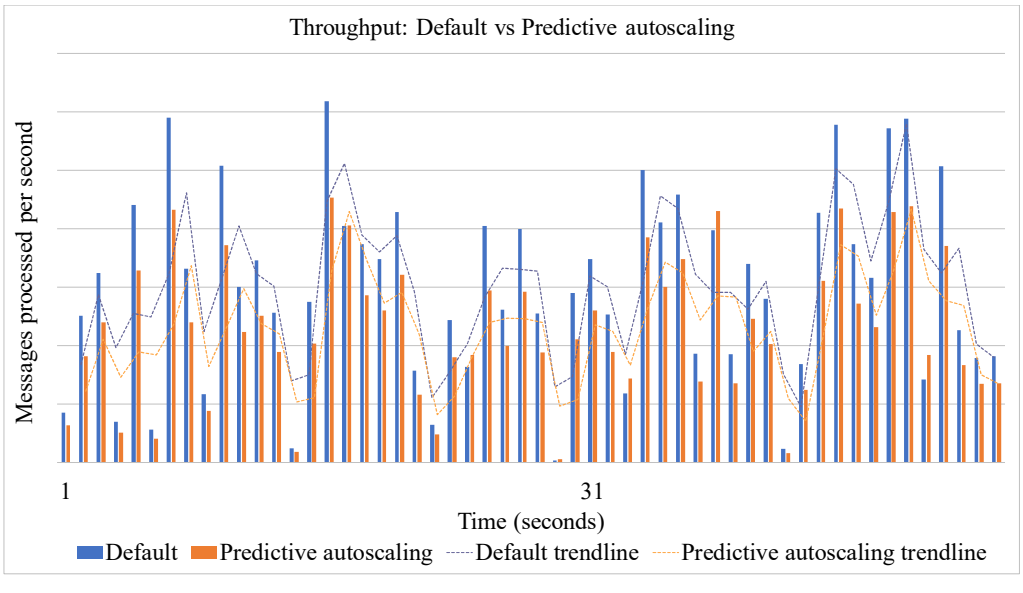

Fig. 4. Throughput: default and predictive autoscaling

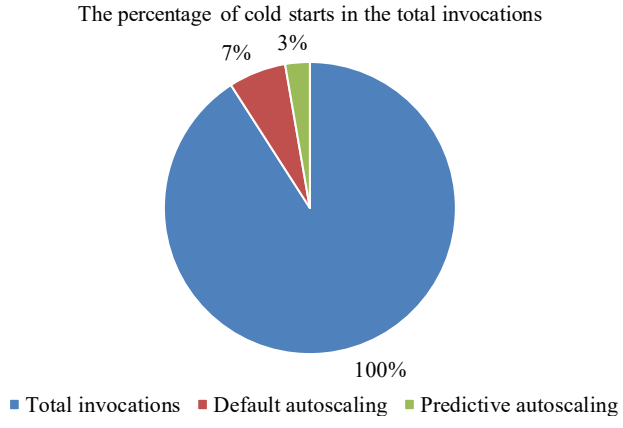

Fig. 5. Percentage of cold starts in total calls

6. Discussion of results from investigating the use of a serverless predictive autoscaling framework for distributed data processing

During our study, the architecture of the framework was designed with the possibility of predictive autoscaling of computing resources for distributed data processing (Fig. 1, 2). Attention focused on the isolation of individual components of the framework to improve the scalability of the system and simplify its maintenance. All components of the framework are serverless and require minimal intervention by developers.

The implementation of predictive autoscaling as a separate module in the framework makes it possible to adapt to peak loads and change the settings of queues and data processors based on the forecast and predefined threshold values.

Using the framework to develop prototypes of serverless applications significantly accelerated data processing, by approximately 25.8% (Fig. 3). This is primarily due to the ability to allocate or release computing resources in a timely manner under varying load conditions. Unlike conventional methods [5, 6], where scaling occurs with a certain delay, our framework improves this process, which is critically important for serverless solutions.

As can be seen from Fig. 4, the system throughput increased by approximately 21.3%. Unlike statically configured applications [7], changes in the configuration of queue and data handler parameters directly during execution make it possible to respond to peaks in load in a timely manner and reduce the time between function calls. Another aspect that has a positive impact on the system throughput is the use of a multi-tiered architecture of the application and message queues. This approach makes it possible to implement weak coupling between individual elements of the system and scale them independently if necessary.

To reduce the number of cold starts of Lambda functions, various techniques are studied in works [8, 23–25], such as periodic warming up of data handlers according to a certain schedule, applying optimizations to reduce initialization time, using caching, allocating reserved instances of handlers, using machine learning, etc. Unlike works [23, 24], which propose a hybrid mechanism for warming up handler functions based on patterns in historical data and using classifiers to identify and launch critical functions, the proposed approach (predictive autoscaling) reduced the proportion of cold starts to 3% (Fig. 5). Our result is comparable to the achievements of hybrid methods. The proposed framework is a complement and extension to the above-mentioned studies in the context of using serverless computing and, unlike them, makes it possible to reduce computational complexity, as well as it does not require the availability of historical data.

The results of our study allow us to state that the main goal, namely, the design of a framework for distributed data processing using predictive automatic scaling of computing resources, has been achieved. The problem of preventive scaling of resources to improve the most critical system indicators has been partially solved. A feature of the developed framework is the ability to be part of any serverless application for distributed data processing using queues.

A limitation of our study that may affect the results is the use of tasks of the same type when testing the designed system. This approach makes it possible to compare the effectiveness of different autoscaling methods but does not fully reflect the actual conditions of functioning of serverless distributed applications. It should be noted that in real applications, tasks can differ significantly in the amount of input data and processing time, which could cause an uneven load on the system and may slightly reduce the efficiency of autoscaling.

The disadvantages of the proposed framework are several external factors that may have a negative impact on the efficiency of autoscaling. Thus, AWS Lambda functions have execution time limitations, making them unsuitable for long-running data processing operations. Using semi-Markov models can increase computational complexity and overhead. The number of states in a model affects the accuracy of the prediction and requires careful tuning.

This implies further evolution of the framework and its adaptation to work with unevenly distributed tasks. It is also worth exploring the possibility of prioritizing tasks based on their size and execution time. This would reduce latency and make the application more stable.

In order to refine forecasting results, a promising area of research is to combine predictive autoscaling with reactive autoscaling in the form of hybrid models. Substantiating the economic feasibility of using the developed framework in real applications and compiling recommendations to avoid excessive allocation of computing resources, especially during periods of low load or in transient phases of load changes is another potential avenue of further research.

7. Conclusions

1. A framework architecture for distributed computing in serverless applications has been designed that uses predictive autoscaling and monitors the processing in real time. A feature of this architecture is the use of a message queue, implementation of weak coupling between individual system elements, their independent scaling if necessary, fast recovery from failures, and optimization of resource usage.

2. A predictive autoscaling module has been implemented that receives information about input data and estimates the probability of the system transitioning between different load states. Semi-Markov models deployed in Amazon SageMaker were used to predict the load. Based on the forecasts, the main parameters of queues and processors were adjusted to optimize resource usage. A feature of the implemented solution is its ability not only to predict changes in load but also initiate the scaling process with minimal delay.

3. The effectiveness of the proposed framework was compared with conventional autoscaling mechanisms. The test results demonstrated an increase in data processing speed by

approximately 25.8%, a decrease in the proportion of cold starts to 3%, and an increase in system throughput by approximately 21.3%. Compared with traditional approaches, the proposed solution works more stably when the load changes. This is explained by the use of predictive scaling, which makes it possible to optimize the configuration of individual system elements in advance before peak values appear.

Conflicts of interest

The authors declare that they have no conflicts of interest in relation to the current study, including financial, personal, authorship, or any other, that could affect the study, as well as the results reported in this paper.

Funding

The study was conducted without financial support.

Data availability

All data are available, either in numerical or graphical form, in the main text of the manuscript.

Use of artificial intelligence

The authors confirm that they did not use artificial intelligence technologies when creating the current work.

References

1. Thumala, S. (2020). Building Highly Resilient Architectures in the Cloud. Nanotechnology Perceptions, 16 (2), 264–284.
2. Mampage, A., Karunasekera, S., Buyya, R. (2025). A deep reinforcement learning based algorithm for time and cost optimized scaling of serverless applications. Future Generation Computer Systems, 173, 107873. https://doi.org/10.1016/j.future.2025.107873
3. Sohani, M., Jain, S. C. (2021). A Predictive Priority-Based Dynamic Resource Provisioning Scheme With Load Balancing in Heterogeneous Cloud Computing. IEEE Access, 9, 62653–62664. https://doi.org/10.1109/access.2021.3074833
4. Beikzadeh Abbasi, F., Rezaee, A., Adabi, S., Movaghar, A. (2023). Fault-tolerant scheduling of graph-based loads on fog/cloud environments with multi-level queues and LSTM-based workload prediction. Computer Networks, 235, 109964. https://doi.org/10.1016/j.comnet.2023.109964
5. Ghorbian, M., Ghobaei-Arani, M. (2024). A survey on the cold start latency approaches in serverless computing: an optimization-based perspective. Computing, 106 (11), 3755–3809. https://doi.org/10.1007/s00607-024-01335-5
6. Tari, M., Ghobaei-Arani, M., Pouramini, J., Ghorbian, M. (2024). Auto-scaling mechanisms in serverless computing: A comprehensive review. Computer Science Review, 53, 100650. https://doi.org/10.1016/j.cosrev.2024.100650
7. Manchana, R. (2020). Operationalizing Batch Workloads in the Cloud with Case Studies. International Journal of Science and Research (IJSR), 9 (7), 2031–2041. https://doi.org/10.21275/sr24820052154
8. Alharthi, S., Alshamsi, A., Alseiari, A., Alwarafy, A. (2024). Auto-Scaling Techniques in Cloud Computing: Issues and Research Directions. Sensors, 24 (17), 5551. https://doi.org/10.3390/s24175551
9. Taha, M. B., Sanjalawe, Y., Al-Daraiseh, A., Fraihat, S., Al-E'mari, S. R. (2024). Proactive Auto-Scaling for Service Function Chains in Cloud Computing Based on Deep Learning. IEEE Access, 12, 38575–38593. https://doi.org/10.1109/access.2024.3375772
10. Verma, S., Bala, A. (2021). Auto-scaling techniques for IoT-based cloud applications: a review. Cluster Computing, 24 (3), 2425–2459. https://doi.org/10.1007/s10586-021-03265-9
11. Kyrychenko, O. (2024). Real-time communication tools for web applications in a cloud environment. The 13th International Conference on Electronics, Communications and Computing's (IC ECCO), 127–128. Available at: https://ecco.utm.md/wp-content/uploads/2024/12/IC-ECCO-2024-AbstractBookBN.pdf
12. Nastic, S., Rausch, T., Scekic, O., Dustdar, S., Gusev, M., Koteska, B. et al. (2017). A Serverless Real-Time Data Analytics Platform for Edge Computing. IEEE Internet Computing, 21 (4), 64–71. https://doi.org/10.1109/mic.2017.2911430
13. Kaur, N., Mittal, A. (2021). Fog Computing Serverless Architecture for Real Time Unpredictable Traffic. IOP Conference Series: Materials Science and Engineering, 1022 (1), 012026. https://doi.org/10.1088/1757-899x/1022/1/012026
14. Amazon Simple Storage Service Documentation. Amazon Web Services. Available at: https://docs.aws.amazon.com/s3
15. Amazon Simple Queue Service. Amazon Web Services. Available at: https://aws.amazon.com/sqs
16. Amazon Web Services. (2025). What is AWS Lambda? Available at: https://docs.aws.amazon.com/lambda/latest/dg/welcome.html
17. Amazon Web Services. (2025). Amazon DynamoDB. Available at: https://aws.amazon.com/dynamodb/.
18. What is AWS AppSync? Amazon Web Services. Available at: https://docs.aws.amazon.com/appsync/latest/devguide/what-is-appsync.html
19. Amazon SageMaker. Amazon Web Services. Available at: https://aws.amazon.com/sagemaker/
20. Amazon EventBridge. Amazon Web Services. Available at: https://aws.amazon.com/eventbridge/
21. Rojas, L., Yepes, V., Garcia, J. (2025). Complex Dynamics and Intelligent Control: Advances, Challenges, and Applications in Mining and Industrial Processes. Mathematics, 13 (6), 961. https://doi.org/10.3390/math13060961
22. Kyrychenko, O. O., Ostapov, S. E., Kyrychenko, O. L. (2025). Optimization of SQS Configurations for Efficient Batch Data Processing. WSEAS Transactions On Systems, 24, 36–43. Portico. https://doi.org/10.37394/23202.2025.24.4
23. Golec, M., Walia, G. K., Kumar, M., Cuadrado, F., Gill, S. S., Uhlig, S. (2024). Cold Start Latency in Serverless Computing: A Systematic Review, Taxonomy, and Future Directions. ACM Computing Surveys, 57 (3), 1–36. https://doi.org/10.1145/3700875
24. Saravana Kumar, N., Selvakumara Samy, S. (2025). Cold Start Prediction and Provisioning Optimization in Serverless Computing Using Deep Learning. Concurrency and Computation: Practice and Experience, 37 (4-5). https://doi.org/10.1002/cpe.8392
25. Nguyen, T. N. (2024). Holistic cold-start management in serverless computing cloud with deep learning for time series. Future Generation Computer Systems, 153, 312–325. https://doi.org/10.1016/j.future.2023.12.011