UDC 621.382.2
DOI: 10.15587/1729-4061.2023.285514

PERFORMANCE EVALUATION OF THE CLOUD COMPUTING APPLICATION FOR IOT-BASED PUBLIC TRANSPORT SYSTEMS

Ihor Zakutynskyi
Postgraduate Student*
Leonid Sibruk
Doctor of Technical Sciences, Professor*
Ihor Rabodzei
Corresponding author
Department of Information Technology Security
National Aviation University
Liubomyra Huzara ave., 1, Kyiv, Ukraine, 03058
E-mail: igor.rabodzei@gmail.com
*Department of Radio-Electronic Devices and Systems National Aviation University
Liubomyra Huzara ave., 1, Kyiv, Ukraine, 03058

The object of research is cloud computing as an element of the server infrastructure for intelligent public transport systems. Given the increasing complexity and requirements for modern transportation, the application of the Internet of Things concept has a high potential to improve efficiency and passenger comfort. Since the load generated in IoT systems is dynamic and difficult to predict, the use of traditional infrastructure with dedicated servers is suboptimal. This study considers the use of cloud computing as the main server infrastructure for the above systems. The paper investigates the main cloud platforms that can be used to develop such systems and evaluates their advantages and disadvantages. The authors developed the overall architecture of the system and evaluated the performance and scalability of individual components of the server infrastructure. To test the system, a software emulator was developed that simulates the controller module installed in vehicles. Using the developed emulator, stress tests were conducted to analyze and confirm the ability to scale and process input data by the proposed architecture. The test scenarios were developed and conducted on the basis of the existing public transportation system in Kyiv, Ukraine. The experimental results showed that the proposed IoT architecture is able to scale efficiently according to the load generated by the connected devices. It has been found that when the number of incoming messages increases from 40 to 6000, the average message processing time remains unchanged, and the error rate does not increase, which is an indicator of stable system operation. The obtained results can be used in the development of modern public transport systems, as well as for the modernization of existing ones

Keywords: internet of things, cloud computing, system architecture, public transport systems, scalability

Received date 17.03.2023
Accepted date 23.05.2023
Published date 30.08.2023

How to Cite: *Zakutynskyi, I., Sibruk, L., Rabodzei, I. (2023). Performance evaluation of the cloud computing application for iot-based public transport systems. Eastern-European Journal of Enterprise Technologies, 4 (9 (124)), 6–13. doi: https://doi.org/10.15587/1729-4061.2023.285514*

1. Introduction

The International Data Corporation (IDC) predicts that the global Internet of Things (IoT) market will grow rapidly, with the number of connected devices reaching 55.7 billion by 2025 [1]. The increase in the number of connected IoT devices will contribute to the creation and development of intellectual public transport systems (IPTS) based on the concept of the Internet of Things.

The integration of IoT solutions into public transport systems has revolutionized the way people move around cities. These systems collect real-time data from buses, trams, trains, and other transport units, allowing operators to optimize routes, improve passenger convenience, and reduce transport costs. Existing intelligent transport systems usually use the classic IoT architecture. With this approach, the data collected by the sensors installed in the transport units are transmitted over the network to the server for further processing and storage. Most existing IoT systems use dedicated servers to store and process data.

Managing and analyzing this vast amount of data requires significant computing power and memory that traditional on-premises infrastructure may not be able to handle. In addition, local solutions are inflexible and unprofitable due to the inability to quickly respond to changes in load. The load generated by IoT devices is dynamic and non-deterministic and can be high (during peak times) or low (during idle times). In both cases, the server infrastructure must adapt to the load generated by IoT clients. The system should work stably with a large number of simultaneously connected devices, and should not use redundant resources with a small number of connected clients. Therefore, there is a need to explore alternatives to traditional methods of dedicated servers to manage the complexity and scalability of the above systems.

The main paradigm of modern cloud technologies is serverless computing. Serverless computing is a model in which a provider dynamically manages and allocates resources to run applications. With serverless computing, the underlying infrastructure automatically scales up or down based on demand. This ensures that the system can handle fluctuations in data volume and traffic without the need for manual configuration or infrastructure management. Usually, the architecture of the IoT system is event-driven, when software functions are triggered based on the arrival of sensor data or as a result of direct interaction with the user (commands). Serverless computing is well-suited to such architectures, as it allows the corresponding functions to be started and executed when a command is received, and to terminate the command immediately after execution. Therefore, research aimed at the development and analysis of IoT systems using serverless computing and cloud technologies is relevant.

2. Literature review and problem statement

In work [2], the general paradigms of server computing and the role of cloud technologies for the concept of the Internet of Things are considered. The reported research results show that the potential of the modern IoT system cannot be fully realized on the basis of classic infrastructures with dedicated servers. This is due to issues with scalability, network bandwidth, and inefficient use of computing resources. These shortcomings are especially evident in systems with dynamic load, such as intelligent transport systems. In [3], the authors consider the concepts of high-level frameworks for intelligent transport systems. In [4], the authors also propose a high-level concept of the system, without analyzing communication technologies and data processing methods. Work [5] proposed a practical implementation of the system using cloud storage and a database. The shortcomings of this work include the fact that it was not tested, and the load resistance of such an architecture was not confirmed. The authors of study [6] proposed an approach for analyzing and processing a large volume of data generated in modern IoT-based transport systems. In the proposed approach, cloud computing is used as the main server infrastructure, but the feasibility of such an approach is not proven. In [7], a system was developed to detect driver drowsiness using neural network methods. For data processing, cloud technologies are also used to save data, as well as to ensure data exchange in real time. Works [2–7] explore the possibilities of integrating cloud technologies into existing transport systems, emphasizing their potential to increase efficiency and analytics.

An important aspect when building an IoT system is the provision of secure communication channels and data security. In the last few years, many studies have appeared that shed light on this topic. In particular, in [8], the authors conduct a security study using modeling at the physical and network levels. The security modeling method proposed in the cited paper can be applied in the construction of the IoT network architecture, in particular, in the design of an intelligent transport system. In [9], a comprehensive study of cyber security issues in cloud computing based on the concept of the Internet of Things is conducted. The authors consider the problems and vulnerabilities that arise as a result of the introduction of cloud computing in IoT systems, and also identify future research directions. Also, a similar study is conducted in [10], where the authors analyze potential attacks in IoT systems, as well as known countermeasures. These studies examine security protocols, encryption methods, and access control mechanisms to address security issues and protect sensitive data. At the same time, the above studies show a number of unresolved problems in the field of data security.

Another unresolved issue is scaling and efficient use of computing resources. The first step in overcoming these difficulties is to identify and analyze the highly loaded components of the system. This is the approach used in work [11], where the general architecture of the public transport system is given, with a complete analysis of communication technologies and data transmission protocols. However, study [11] does not consider the behavior of the system under dynamically changing load, and therefore its ability to scale.

This gives reason to assert that it is appropriate to conduct a study to analyze the use of serverless computing methods in Internet of Things systems in general and in the context of intelligent transport systems.

3. The aim and objectives of the study

The purpose of this study is to determine the possibilities of serverless computing in intelligent transport systems based on the Internet of Things. This will provide an opportunity to evaluate the effectiveness and feasibility of using serverless cloud computing as the main server infrastructure for data processing and storage.

To achieve the goal, the following tasks were set:

– to develop a general software architecture for processing and saving data of an intelligent transport system;

– to develop a prototype and emulator of a car module for data generation, testing and determination of system limits;

– to investigate the possibilities of scaling, as well as the efficiency of the system under load.

4. The study materials and methods

4. 1. The object and hypothesis of the study

The object of research is serverless cloud computing as an element of the server infrastructure for intelligent public transport systems.

The hypothesis of the study assumed that serverless cloud could significantly increase the efficiency and speed of server data processing, based on methods of dynamic scaling of resources.

The research was carried out using prototypes, an emulator of a car module (IoT controller), and a deployed server infrastructure on the AWS IoT Core platform.

In this study, the surface urban transport system of Kyiv, Ukraine was taken as a basis; load modeling and forecasting were carried out according to these data. Public ground transport in Kyiv consists of buses, trolleybuses, and trams, which are used by an average of 1.1 million residents every day [12] (Table 1).

Table 1

State of public transport in Kyiv, 2019

Type of transport	Routes	Number of vehicles
Buses	105	400
Trolleybuses	50	370
Trams	20	290

In the research process, it was assumed that all transport units would be on the route at the same time, and the test load would be planned according to this amount of data and transmission speed.

4. 2. System requirements

In the structure of the IoT system, the controller is the main source of data that transmits and receives information from the server through the software level protocol – MQTT (Message Queue Telemetry Transport). Table 2 lists the main data types and MQTT sections used to exchange messages between the device and the server.

Table 2

MQTT sections

Section	Size (bytes)	Interval (seconds)	Bandwidth	Data description
geolocation	100	5	12	Current coordinates
fuel_level	30	10	6	Fuel level/ Battery charge
climate/temperature	30	60	1	Temperature
climate/humidity	30	60	1	Humidity, %
passengers_count	30	60 seconds after the Stop event	Max. 1	Number of passengers

Table 3 [13] shows the accessibility requirements of modern cloud systems according to the type of system.

Table 3

Accessibility requirements

Availability	Unavailability (hours per year)	System type
99 %	3 days 15 hours	Batch processing, data transfer and download
99.9 %	8 days 45 minutes	Monitoring
99.95 %	4 hours 22 minutes	E-commerce platforms
99.99 %	52 minutes	Video services
99.999 %	5 minutes	Payment transactions, telecommunications loads

Since the system generates dynamic and non-deterministic data flows, the cloud part must provide the highest class of availability – 99.999 %.

The number of connected (active) devices/clients can constantly change, so the system must scale according to the load.

Data transmitted between the device and the cloud must be securely protected. In addition, it is necessary to limit user access to the system by roles and access level.

The system must provide a cost-effective way to store and process data generated by IoT devices. Traditional server architectures require dedicated servers even during periods of low usage. Serverless computing makes it possible to pay only for the actual use of resources since the cloud provider manages the infrastructure and dynamically allocates resources. This pay-as-you-go model can lead to cost savings, especially in IoT systems where the load can vary greatly at different points in time.

4. 3. Research methodology

The methodology of test development (TDD – test driven development) was used to conduct the research. TDD is a software development process that involves creating software requirements through the description of software tests before the software is fully developed. This approach differs from the conventional method of developing the software first and then describing the tests.

According to the TDD methodology (Fig. 1), an emulator of the IoT module was designed.

The developed emulator repeats the algorithms of the car module (Fig. 2) as well as the data structure transmitted to the server.

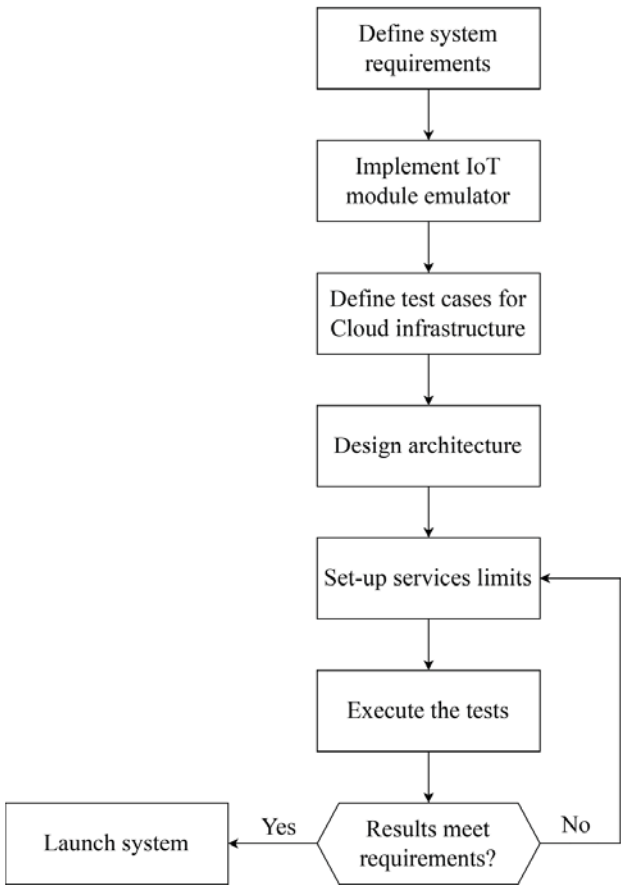

Define system requirements
Implement IoT module emulator
Define test cases for Cloud infrastructure
Design architecture
Set-up services limits
Execute the tests
Results meet requirements?
Yes
No
Launch system

Fig. 1. Algorithm of the study

```
Algorithm 1: Automotive controller firmware
while Network is not connected do
    Init Nb-IoT connection...
end
Connect to MQTT Gateway...
if mqtt connection then
    Subscribe to MQTT topics → {geolocation, climate, fuel, passengers}
end
while true do
    if intervalEvent then
        Read / Publish data → {geolocation, climate, fuel} topic
    end
    if stopEvent then
        if stopLocation in (currentPosition + precisionDistance) region then
            Listen passengers module delay(60s) then
            Publish data → passengers topic
        end
    end
end
```

Fig. 2. The algorithm of the automotive module

The emulator can be run programmatically with any level of parallelism, which makes it possible to define system limits on the number of connected devices.

5. Results of research into the application of cloud computing for the intelligent system of public transport

5. 1. Development of the general system architecture for data processing

Based on the analysis of existing cloud platforms for Internet of Things systems, AWS IoT Core was taken as a basis. This platform fully meets the set technical requirements, communication security requirements, and also has the most distributed infrastructure.

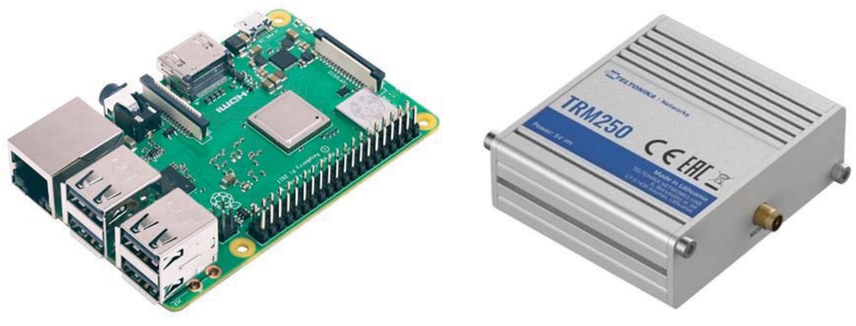
Fig. 3 shows the general architecture of the system, which is conventionally divided into the following components (groups): a group of automotive modules, an Internet of Things gateway, data processing and storage.

An additional advantage of the AWS IoT platform is the pay-per-second computing model, which makes this approach cost-effective, as only the resources used (computing hours) are paid for.

5. 2. Design of a device prototype and emulator to analyze and evaluate the performance of the proposed architecture

A group of automotive modules is a group of Internet of Things devices that are installed in transport units and transmit sensor data to a server. Each device consists of a controller, an LTE modem, and connected sensors. The device controller reads data from sensors and forms packets for transmission. Data is transmitted using NB-IoT (Narrow Band Internet of Things) technology via an LTE channel. MQTT is used as a software layer protocol.

Fig. 4 shows a prototype of a car module based on a Raspberry PI microcomputer and a Teltonika TRM250 modem.



a *b*

Fig. 4. Automotive module prototype: *a* – Raspberry Pi [14]; *b* – Teltonika TRM 250 [15]

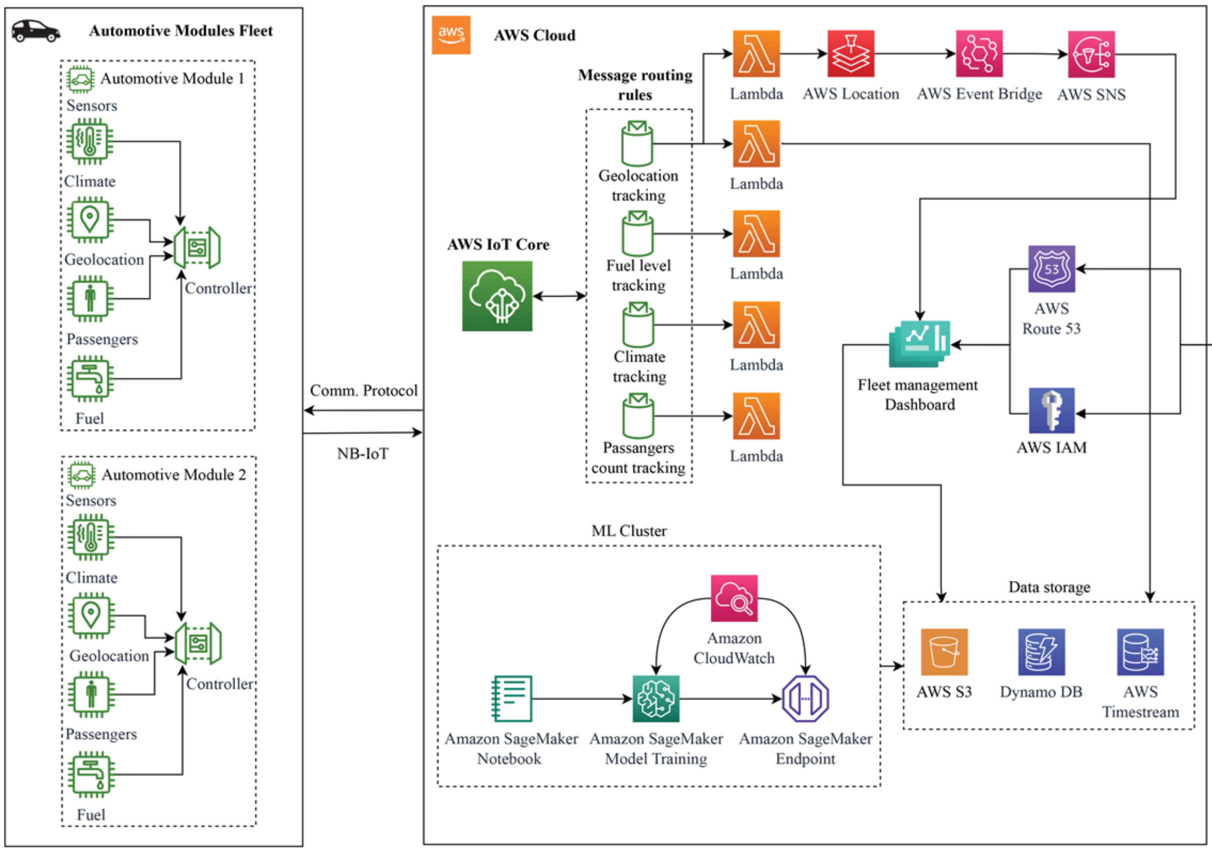


Fig. 3. General system architecture

Fig. 5 shows the diagram of information flows between the client (automotive module) and the server (AWS IoT Core).

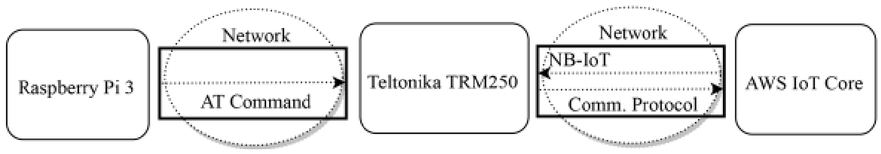


Fig. 5. Scheme of information flows of the automotive module

In this case, the automotive module is both a subscriber and a publisher, according to the concept of the MQTT protocol.

5. 3. Investigating the possibilities of scaling and the efficiency of the data storage and processing system

In the proposed architecture, all received data via the MQTT protocol is distributed on special topics according to routing rules. Routing rules define filtering criteria based on the content of an incoming message and perform the distribution of incoming messages between AWS services. In this case, routing of incoming messages between AWS IoT Core and AWS Lambda will be considered for further processing, analysis, and evaluation of system performance.

AWS Lambda functions are stateless functions for executing software code. In this case, lambda functions are used as the first level of data processing. The Lambda function code is responsible for receiving a message from the device and forming a structure for further storage or transmission to another service. For the proposed system, two cases of using the lambda function are considered:

– forming a data structure and saving it in a database based on time series (geolocation, climate data, fuel level, number of passengers);

– forming the data structure and sending it to the location monitoring service. Lambda functions receive data via the HTTP protocol. Processing functions (Fig. 6) are written using the Python programming language.

The proposed architecture uses three services for data storage: AWS Dynamo DB, AWS S3, and AWS Timestream.

As IoT modules generate large amounts of data in real time, it is necessary to provide time-series-based retention. AWS Timestream is designed to collect, store, and analyze time series data from devices that emit a sequence of time-stamped data [16]. The database allows recording in two different formats: multidimensional, which allows multiple measurements per record, and unidimensional, which allows only one measurement per record. Since the devices of the automotive module transmit different indicators in different periods of time (Table 1), a one-dimensional format was used (Table 4).

Table 4

Unidimensional recordings Timestream

Device ID	Module	Measurement name	Time	Value
auto-1127	kyiv-west	fuel_level	2023-04-15 19:00:01	66.14
auto-2448	kyiv-west	fuel_level	2023-04-15 19:00:01	100
auto-1127	kyiv-west	fuel_level	2023-04-15 19:00:12	66.14
auto-2331	kyiv-west	fuel_level	2023-04-15 19:00:24	42.29
auto-1127	kyiv-west	fuel_level	2023-04-15 19:00:26	66.13

AWS Dynamodb and AWS S3 are used as additional databases to store user preferences, configurations, and artifacts.

```
def lambda_handler(event, context):
  update = {
    DeviceId: event[payload][deviceid],
    SampleTime: datetime.fromtimestamp(event[payload][timestamp]),
    Position: [
        event[payload][location][long],
        event[payload][location][lat]
      ]
  }
  if accuracy in event[payload]:
    update[Accuracy] = event[payload][accuracy]

  if positionProperties in event[payload]:
    update[PositionProperties] = event[payload][positionProperties]

  client = boto3.client(location)

  response = client.batch_update_device_position(
      TrackerName,
      Updates
  )

  return {
    statusCode: 200,
    body: json.dumps(response)
  }
```

Fig. 6. Example of a lambda function

The purpose of testing is to reproduce the load that will be created by the intelligent public transport system of Kyiv (Table 4). It is assumed that each transport unit is equipped with an automotive module that transmits data to the server via the NB-IoT connection, and that all transport units are on the route at the same time. It is necessary to evaluate the performance and scalability of the system according to the change in load: the number of connected devices and the amount of data they generate.

An automotive module emulator was designed for testing. The emulator reproduces the algorithm (Fig. 2), as well as data types and structures (Table 1). For simulation data, the emulator uses a set of geolocation points of the trolleybus route in Kyiv, as well as randomly generated data on temperature, humidity, fuel level, and number of passengers.

Below are the main results and key indicators of the conducted tests. The load was generated for 45 minutes.

The ratio of successful requests – this indicator represents the percentage of successful calls (Fig. 7) of the lambda function during the testing period.

The ratio is calculated by dividing the number of successful calls by the total number of calls (events).

The frequency of function calls is a call indicator that shows the number of calls to the lambda function (Fig. 8) over a certain period of time.

In the context of the proposed system, this means the number of received messages from the IoT Core. This indicator makes it possible to evaluate the volume of incoming messages, as well as their dynamics.

Function execution time – this indicator makes it possible to estimate the amount of time required to execute the logic described in the lambda function (Fig. 9). In this case, it is estimated how much time is spent on receiving messages, building a data structure, and saving information in the database.

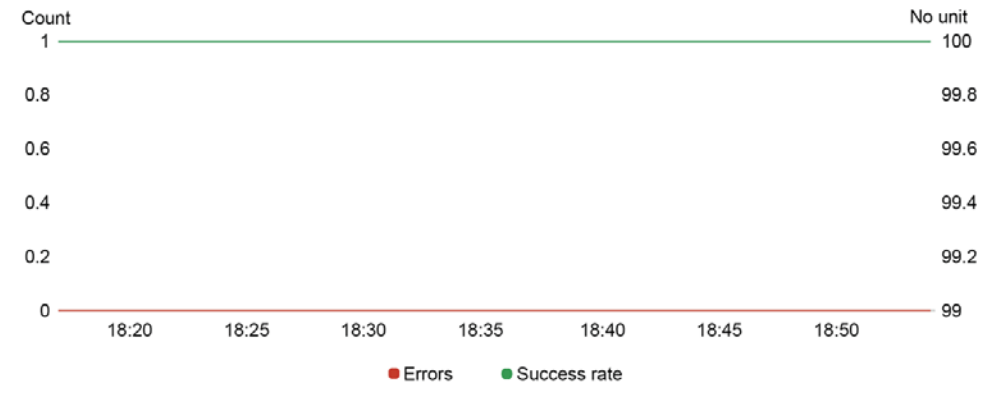


Fig. 7. Success rate of lambda function queries

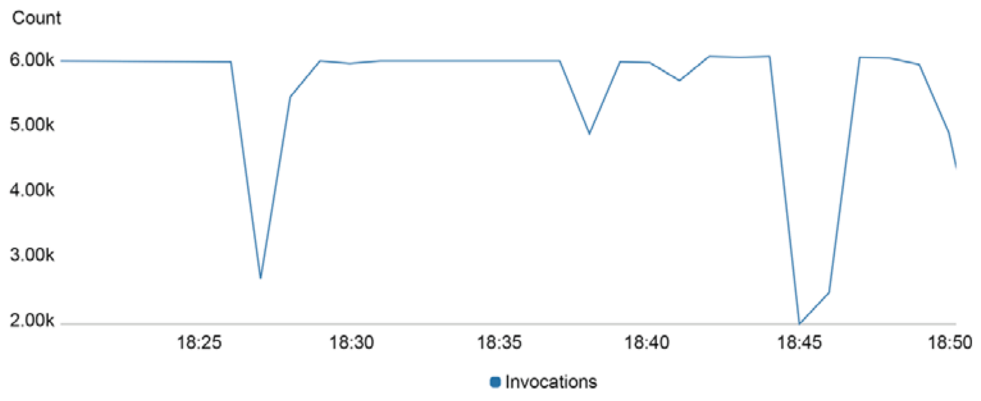


Fig. 8. Lambda function calls

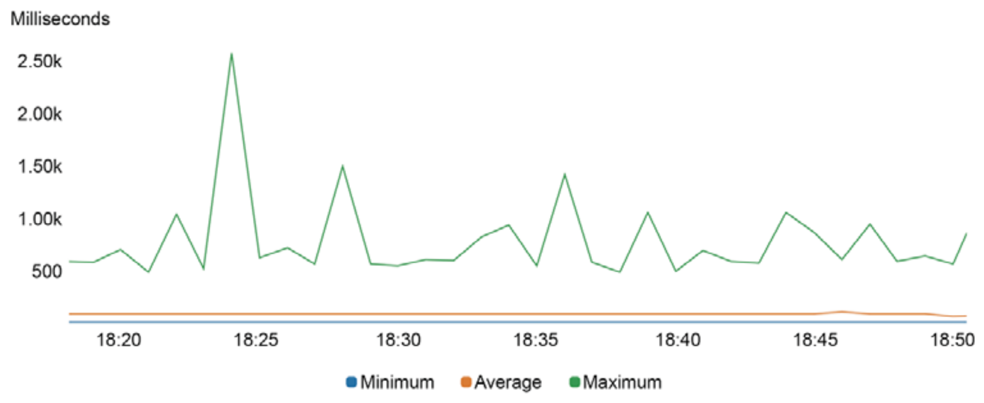


Fig. 9. Function execution time

The number of parallel launches makes it possible to estimate the number of parallel launches of the function (Fig. 10).

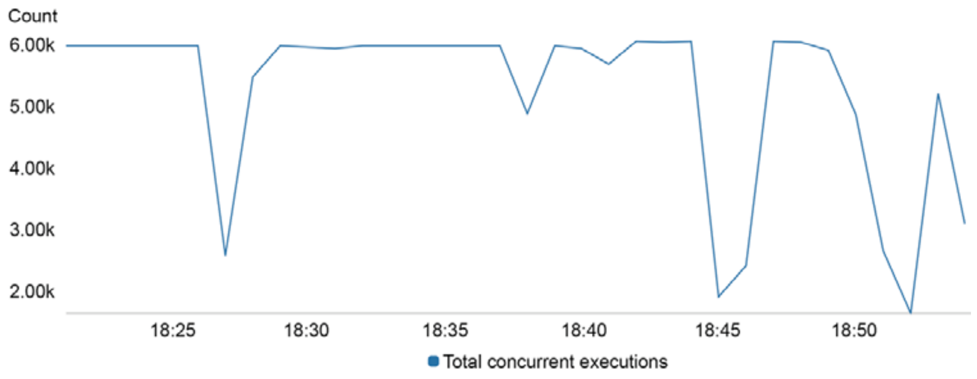


Fig. 10. Number of parallel launches of lambda functions

Since the load is dynamic, there is a need for parallel function calls.

6. Discussion of results of investigating the effectiveness of cloud computing for intelligent public transport systems

Our results demonstrate and confirm the feasibility of using serverless cloud computing for data processing and storage in Internet of Things systems, in particular in the context of intelligent transport systems.

To conduct the experiment, a general cloud system architecture (Fig. 3) is proposed, which allows receiving data from IoT devices in real time, as well as processing and saving them. The system is built on the basis of the concept of microservice architecture, where each logical element is a separate microservice. This approach made it possible to scale each individual microservice according to the load on it, without affecting other components. In the proposed MQTT system, the server is the input gateway for connecting IoT devices.

To reproduce the workload of the system, a prototype of a connected automotive module (Fig. 4) was designed, which transmits data to the server, as well as its software emulator, which fully reproduces the work algorithm and data structures of the IoT device. Effective load testing of the system architecture was carried out on the basis of the proposed IoT controller software emulator. The method of emulating connected devices is extremely effective because it allows one to reduce costs and simplify the validation and testing of the system before its implementation in the real environment, and it also allows the system to be tested at different scales and configurations. In this study, a simulation of the transport system of the city of Kyiv was carried out.

Based on our experiment, the possibilities of scaling the system according to the input load were evaluated. During the entire testing period (Fig. 7), no performance errors were detected. This means that all messages have been delivered and the following data storage and processing logic has been executed successfully. From the plot (Fig. 8) it can be seen that the system simultaneously executed ~6 thousand functions. In addition, it can be seen that changing the number of function calls does not affect the overall rate of successful requests (Fig. 7). Also, the plot (Fig. 9) shows that with an increase in the number of messages, the maximum execution time increases (to 2 seconds) but the average indicator remains unchanged. The lambda function automatically scales resources (Fig. 10) to process incoming requests and messages. As demand grows, additional containers are created, which are automatically removed when the load decreases. The results of performance tests show that the change in load does not significantly affect the main indicators of processing speed and data retention. These results are due to the principles of dynamic management of computing resources inherent in cloud systems, and are the main advantage over traditional infrastructures with dedicated servers.

The proposed system development methodology, as well as the software architecture, can be used to build and modernize modern Internet of Things systems, in particular, intelligent transport systems. Unlike systems with a static server infrastructure [17], the proposed architecture is scalable and can dynamically respond to changes in the input load.

The disadvantages of the proposed technique include:

– dependence on the communication network: to use cloud systems, constant access to a fast and reliable Internet connection is necessary. In the case of failure of the communication network or unavailability of the Internet, there may be a problem with the functioning of the system;

– complexity of development and support.

The limitation of this study is that the construction and analysis of the data processing and storage system was carried out only on the basis of one platform – AWS IoT. Further development could be to analyze and build similar systems on cloud platforms such as GCP, IBM, and Azure to assess differences in performance and architecture.

7. Conclusions

1. In the course of executing the task set, a general software architecture was developed for processing and saving data of an intelligent transport system based on cloud computing. A microservice architecture was used to build the system, where each module is a separate microservice. This approach makes it possible to dynamically manage computing resources (increase or decrease) according to the load. Also, the advantage of this architecture is that if one of the services is unavailable, the system will continue to work.

2. To perform simulation experiments, prototypes of automotive modules (IoT controllers) were designed, as

well as corresponding software emulators that fully reproduce the algorithm and data structures of a real device. This approach made it possible to test the system under conditions close to actual ones, as well as evaluate the ability to scale and the efficiency of using computing resources. In the course of the experiment, a simulation of the intelligent transport system of the city of Kyiv was carried out.

3. The results of our experiments confirm that the proposed architecture is dynamic and scalable since the change in the number of connected devices and the volume of input data do not significantly affect the stability and speed of processing. Based on the research, it can be concluded that cloud computing could be used in intelligent public transport systems as the main server infrastructure. In addition, this approach is flexible and cost-effective compared to traditional dedicated servers.

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 and the results reported in this paper.

Funding

The study was conducted without financial support.

Data availability

All data are available in the main text of the manuscript.

References

1. Future Of Industry Ecosystems: Shared Data And Insights. IDC. URL: https://blogs.idc.com/2021/01/06/future-of-industry-ecosystems-shared-data-and-insights/
2. Mchergui, A., Hajlaoui, R., Moulahi, T., Alabdulatif, A., Lorenz, P. (2023). Steam computing paradigm: Cross-layer solutions over cloud, fog, and edge computing. IET Wireless Sensor Systems. doi: https://doi.org/10.1049/wss2.12051
3. Porru, S., Misso, F. E., Pani, F. E., Repetto, C. (2020). Smart mobility and public transport: Opportunities and challenges in rural and urban areas. Journal of Traffic and Transportation Engineering (English Edition), 7 (1), 88–97. doi: https://doi.org/10.1016/j.jtte.2019.10.002
4. Farkas, K., Feher, G., Benczur, A., Sidlo, C. (2015). Crowdsending based public transport information service in smart cities. IEEE Communications Magazine, 53 (8), 158–165. doi: https://doi.org/10.1109/mcom.2015.7180523
5. Vieira, E., Almeida, J., Ferreira, J., Dias, T., Vieira Silva, A., Moura, L. (2023). A Roadside and Cloud-Based Vehicular Communications Framework for the Provision of C-ITS Services. Information, 14 (3), 153. doi: https://doi.org/10.3390/info14030153
6. Metzger, F., Hobfeld, T., Bauer, A., Kounev, S., Heegaard, P. E. (2019). Modeling of Aggregated IoT Traffic and Its Application to an IoT Cloud. Proceedings of the IEEE, 107 (4), 679–694. doi: https://doi.org/10.1109/jproc.2019.2901578
7. Khan, M. A., Nawaz, T., Khan, U. S., Hamza, A., Rashid, N. (2023). IoT-Based Non-Intrusive Automated Driver Drowsiness Monitoring Framework for Logistics and Public Transport Applications to Enhance Road Safety. IEEE Access, 11, 14385–14397. doi: https://doi.org/10.1109/access.2023.3244008
8. Hind, M., Noura, O., Sanae, M., Abraham, A. (2023). A Comparative Study for Modeling IoT Security Systems. Lecture Notes in Networks and Systems, 258–269. doi: https://doi.org/10.1007/978-3-031-35510-3_25
9. Ahmad, W., Rasool, A., Javed, A. R., Baker, T., Jalil, Z. (2021). Cyber Security in IoT-Based Cloud Computing: A Comprehensive Survey. Electronics, 11 (1), 16. doi: https://doi.org/10.3390/electronics11010016
10. Siwakoti, Y. R., Bhurtel, M., Rawat, D. B., Oest, A., Johnson, R. C. (2023). Advances in IoT Security: Vulnerabilities, Enabled Criminal Services, Attacks, and Countermeasures. IEEE Internet of Things Journal, 10 (13), 11224–11239. doi: https://doi.org/10.1109/jiot.2023.3252594
11. Zakutynskyi, I., Sibruk, L., Kokarieva, A. (2023). IoT System for Monitoring and Managing Public Transport Data. WSEAS TRANSACTIONS ON SYSTEMS, 22, 242–248. doi: https://doi.org/10.37394/23202.2023.22.25
12. Kyivpastrans. Wikipedia. URL: https://en.wikipedia.org/wiki/Kyivpastrans
13. Availability. Amazon. URL: https://docs.aws.amazon.com/wellarchitected/latest/reliability-pillar/availability.html
14. Image "RaspberryPi B3 +". URL: https://media.distrelec.com/Web/WebShopImages/landscape_large/8-/01/RaspberryPi_B3_plus_30109158-01.jpg
15. Image "Teltonika TRM 250". URL: https://wiki.teltonika-networks.com/images/3/3f/Trm250_hd_1.png
16. Data modeling. Amazon. URL: https://docs.aws.amazon.com/timestream/latest/developerguide/data-modeling.html
17. Massaro, A., Selicato, S., Galiano, A. (2020). Predictive Maintenance of Bus Fleet by Intelligent Smart Electronic Board Implementing Artificial Intelligence. IoT, 1 (2), 180–197. doi: https://doi.org/10.3390/iot1020012