International Journal of Science and Research (IJSR)
ISSN: 2319-7064
SJIF (2022): 7.942

Performance Analysis and Optimization of Amazon Web Service Lambda Functions in Serverless Systems

Dr. Ima Hussain

Assistant Professor, Computational Science Department, BrainWare University, Kolkata, West Bengal, India

Abstract: ***This article explores the working model of AWS Lambda functions and provides a step-by-step guide on creating Lambda functions for different use cases. AWS Lambda is a serverless computing service offered by Amazon Web Services (AWS), allowing developers to run code without the need for provisioning or managing servers. This research paper explores the potential of serverless computing, with a focus on building a robust working model utilizing AWS Lambda, a leading serverless computing platform. The paper delves into the key concepts of serverless architecture, highlighting its benefits, challenges, and practical applications. Through a comprehensive examination of AWS Lambda, the research aims to provide insights into designing and implementing efficient serverless solutions. The study includes real-world use cases, best practices, and performance considerations, offering a practical guide for developers and organizations seeking to harness the power of serverless computing in their applications. The study demonstrates that a modified formula for calculating non-zero demand averages significantly improves forecasting accuracy for intermittent demand data in serverless environments.***

Keywords: AWS Lambda function, Serverless function, Amazon EC2.

1. Introduction

AWS Lambda is a powerful and flexible service that simplifies the process of building and deploying applications that can respond to events and scale automatically. By eliminating the need for server management, Lambda enables developers to focus on writing code and delivering value to their users. In the dynamic landscape of contemporary computing, the paradigm shift towards serverless architecture has emerged as a transformative force, redefining how applications are developed, deployed, and scaled. This research paper embarks on a journey to unlock the potential of serverless computing, focusing on the construction of a robust working model using the AWS Lambda platform. As organizations increasingly gravitate towards cloud-native solutions, the allure of serverless computing lies in its promise to liberate developers from the intricacies of infrastructure management, enabling them to concentrate on code logic and application functionality.

Serverless computing, despite its name, does not imply the absence of servers. Instead, it abstracts the underlying infrastructure, allowing developers to execute code in response to events without the need to provision or manage servers explicitly. AWS Lambda, a prominent player in the serverless arena, exemplifies this paradigm by providing a scalable and cost-efficient platform for running code in response to a variety of events, ranging from HTTP requests to database modifications.

This research endeavours to provide a comprehensive exploration of serverless computing, delineating its advantages and addressing the challenges that accompany this innovative approach. By centering the investigation on AWS Lambda, we seek to offer practical insights into building a resilient working model. Real-world use cases, best practices, and considerations for performance optimization will be meticulously examined to furnish developers and organizations with a practical guide for harnessing the full potential of serverless computing within their applications. The evolving landscape of cloud technologies demands a nuanced understanding of the tools and frameworks available, and this research aims to contribute to this understanding by providing a detailed examination of AWS Lambda. By doing so, we aspire to empower developers and decision-makers to make informed choices in their pursuit of scalable, efficient, and cost-effective computing solutions.

The research paper introduces the innovative functionality of AWS Lambda functions in event-driven computing, allowing developers to execute code in response to a diverse range of events It explores the potential of serverless computing, focusing on building a robust working model utilizing AWS Lambda as a leading serverless computing platform

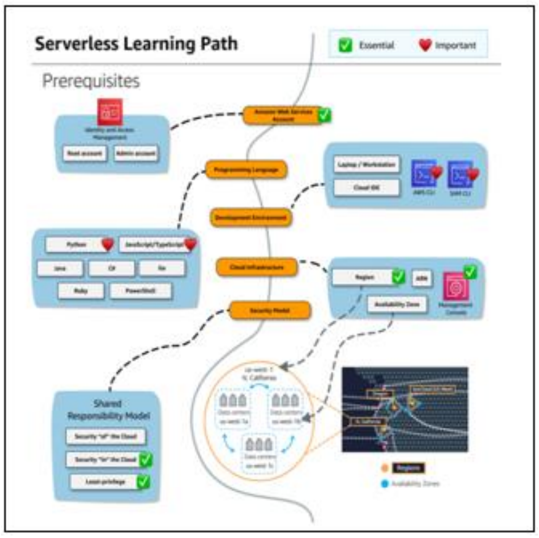

Figure 1

Serverless computing, exemplified by AWS Lambda, has gained widespread adoption for its promise of scalability, cost-efficiency, and reduced operational overhead. This research paper delves into the realm of serverless architectures, specifically focusing on the performance characteristics and optimization strategies of AWS Lambda functions. Through rigorous empirical analysis and experimentation, we scrutinize factors influencing execution times, resource allocation, and scalability in diverse use cases.

1.1 Stateless Execution: Unleash the Freedom

While invoking a function, Lambda ensures a stateless execution environment. This means that your functions do not retain any state between invocations. Instead, your valuable state information can be securely stored in external storage solutions, such as databases. By adopting this approach, you are free from the limitations of storing state within the execution environment. Stateless execution enables you to effortlessly manage your code while conveniently accessing and maintaining critical data.

1.2 Automatic Scaling: Embrace Efficiency and Flexibility

Serverless, like Lambda, are designed to seamlessly scale the number of function instances based on demand they encounter. This automatic scaling not only optimizes resource utilization but also empowers your applications to gracefully handle varying workloads. Gone are the days of worrying about managing server capacity or provisioning resources. Thanks to Lambda's automatic scaling, your application remains agile and resilient, regardless of incoming traffic fluctuations. (S.Narula)

1.3 Micro-Billing Model-

In the realm of serverless computing, traditional capacity allocation gives way to a more practical and cost-effective approach. Billing is closely tied to the actual usage of resources, rather than being confined to pre-allocated capacity. Users are invoiced based on the number of function executions as well as the time taken to execute each function. Embracing this micro-billing model ensures that you only pay for what you genuinely consume, keeping your costs in check while optimizing your budget.

1.4 No Server Management:

AWS Cloud takes charge of server management, provisioning, maintenance, and scaling under the hood, leaving you free to focus on crafting remarkable applications. With Lambda as your ally, you can effortlessly scale your application in response to incoming traffic without breaking a sweat. Embrace the power of no-server management and unlock a whole new level of development freedom!

1.5 Fast Deployment-

In the world of serverless functions, speed is the name of the game. With rapid deployment capabilities, you can swiftly bring your functions to life, revolutionizing your development and deployment cycles. This exceptional agility proves invaluable, especially for applications faced with dynamic and shifting requirements. So, gear up and embrace fast deployment—with Lambda by your side, you'll always stay ahead of the curve.

1.6 Scalability

Serverless architectures are tailored for scalability, empowering your applications to effortlessly adapt to varying workloads. The flexibility and responsiveness of functions allow for instant scalability in response to fluctuations in demand. No matter how unpredictable the surges or dips in traffic may be, Lambda ensures that your applications soar above the challenges, delivering remarkable performance and seamless user experiences. (P.R.Brenner)

1.7 Ephemeral Compute

In the world of serverless computing, functions come and go like a passing breeze. These ephemeral beings are created sponsor to events and gracefully exit the stage once their execution is complete. This ephemeral compute design optimizes resource utilization and ensures efficient usage of computing power. So let this transformative approach redefine efficiency and breathe life into your serverless applications.

1.8 Integrated Services

Serverless platforms, like Lambda, don't operate in isolation. They seamlessly integrate with a vast array of cloud services, enabling you to tap into the full potential of databases, storage solutions, authentication mechanisms, and more. The beauty lies in the effortless connectivity—no additional configuration headaches. With Lambda as your trusty companion, harness the power of integration and elevate your applications to new heights.

2. Literature Review

This paper [1] gives an overview on cloud computing security. To clarify cloud security, a definition and scope of cloud computing security is presented. An ecosystem of cloud security is shown to illustrate what each role in industry can do in turn. Then security impacts of cloud security for both customers and operators are analyzed. [2] this research is supported by a case study of a serverless chat application built using AWS Lambda and other AWS cloud services. The architecture of the messenger application requires very less management and is cost efficient. [3] Implementing the messenger application using the above-mentioned technologies helped in grasping the usefulness and relevance of serverless computing and 'FaaS' in hosting dynamic applications with lots of user interaction. [4]performed a survey for considering limitations and problems of previous research in the research area to find out the most challenging issue in access control and user authentication algorithms. [5] For security and verification numerous procedures given by AWS were found that covers the utilize of sign on session. For verification OTP was found more solid and secure [6]. Cloud suggestions relating to foundation versatility, stack adjusting, provisioning variety and memory reservation were moreover considered [7]. This study tells the engineering of serverless computing and its proficiency. Cross server information administration, asset allotment and confinement among other thing[8]. Diverse systems that cater to a certain basis of real-life application were too examined. The comparison between different cloud stage was carried out [9]. [10] Intelligently employments of different amazon cloud admin is trations were investigated [11]. The assessment of the practicality of the administrations advertised by AWS was checked [12]. The major reason for the move of worldview to serverless cloud computing application was the foundation taken a toll comparison of running a web application on AWS lambda. This paper [13] appropriately compared the major cloud administrations models were IaaS, SaaS and PaaS but with the coming of serverless computing a modern cloud benefit show is presented FaaS (Work as a Benefit) [15] [16]

3. Proposed Design

In order to create a working model of an AWS Lambda function, there are several steps that need to be followed. An example of using the AWS Management Console to create a Lambda function in Python that responds to an Amazon S3 event.

Steps:

1) **Login to AWS Console:** Begin by logging in to the AWS Management Console.
2) **Open Lambda Service:** Once logged in, navigate to the Lambda service.
3) **Create Function:** Click on the "Create function" button to start creating your Lambda function.
4) **Configure Function:**
   - **Name your function:** Give your Lambda function a name that is meaningful to you (e.g., "MyS3Function").
   - **Choose a runtime:** Select the appropriate runtime for your function (e.g., Python 3.8).
   - **Role:** Choose an existing role with S3 and Lambda permissions, or create a new role.
   - **Click "Create function":** Once you have provided the necessary details, click on the "Create function" button to proceed.

a) **Add Trigger:**
- In the designer section, click on "Add Trigger" to set up a trigger for your Lambda function.
- Choose "S3" from the list of services.
- Configure the S3 trigger with the specific bucket and event type you want the function to respond to (e.g., "All object create events").
- Click "Add" to add the trigger to your function.

b) **Write Code:** In the "Function code" section, you can now write your Python code for the Lambda function.

Web application – It Combines the Lambda with other AWS administrations to construct capable web applications that consequently scale up and down and run in a exceedingly accessible setup over different information centres. Underneath graph appear the usefulness of Amazon S3, API Door, AWS Lambda, and DynamoDB that work together to recover climate information for a web or versatile application.

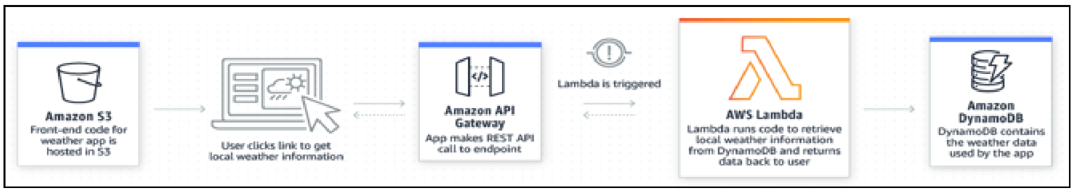
Amazon S3
Front-end code for weather app is hosted in S3
User clicks link to get local weather information
Amazon API Gateway
App makes REST API call to endpoint
Lambda is triggered
AWS Lambda
Lambda runs code to retrieve local weather information from DynamoDB and returns data back to user
Amazon DynamoDB
DynamoDB contains the weather data used by the app

Figure 2: Web Application

IoT Backend - It builds serverless backends utilizing AWS Lambda to handle web, versatile, Web of Things (IoT), and third-party API requests.

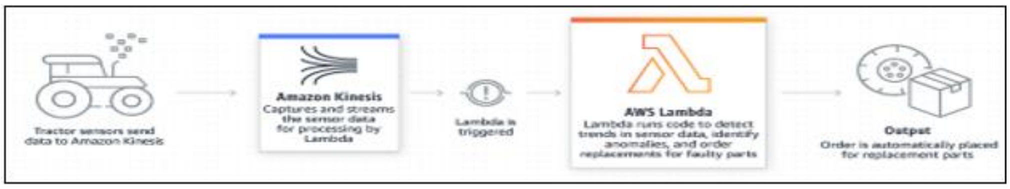
Tractor sensors send data to Amazon Kinesis
Amazon Kinesis
Captures and streams the sensor data for processing by Lambda
Lambda is triggered
AWS Lambda
Lambda runs code to detect trends in sensor data, identify anomalies, and order replacements for faulty parts
Output
Order is automatically placed for replacement parts

Figure 3: IoT Backend

Web Backend -Diagram appearing how Amazon API Portal, AWS Lambda, and Amazon SNS work together to assist clients get status upgrades notices in a portable application.

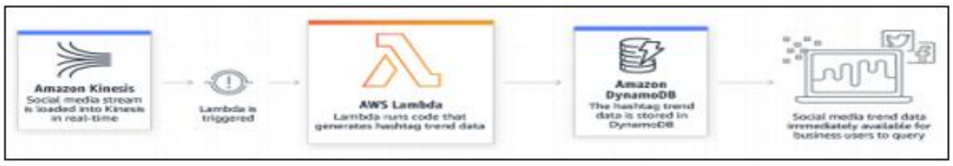
Amazon Kinesis
Social media stream is loaded into Kinesis in real-time
Lambda is triggered
AWS Lambda
Lambda runs code that generates hashtag trend data
Amazon DynamoDB
The hashtag trend data is stored in DynamoDB
Social media trend data immediately available for business users to query

Figure 4: Web Backend

Mobile backends – They are utilizing Lambda and Amazon API Portal to confirm and prepare API demands. Utilize AWS Amplify to easily coordinated along with your iOS, Android, Web, and Respond Local frontends. When utilizing Lambda, you're mindful as it were for your code. Lambda oversees the compute armada that gives a adjust of memory, CPU, organize, and other assets to run your code. Since Lambda oversees these assets, you cannot log in to compute occasions or customize the working framework on given runtimes. Lambda performs operational and regulatory exercises on your sake, counting overseeing capacity, observing, and logging your Lambda functions

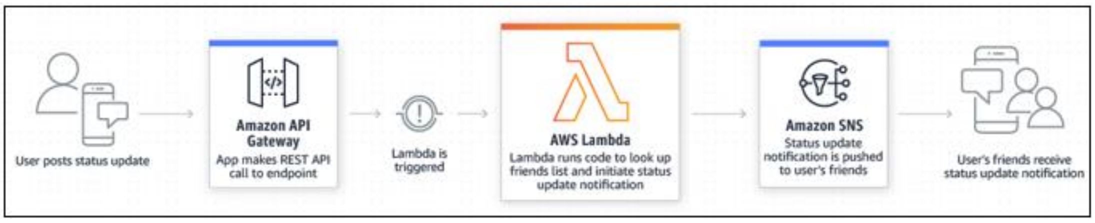
User posts status update
Amazon API Gateway
App makes REST API call to endpoint
Lambda is triggered
AWS Lambda
Lambda runs code to look up friends list and initiate status update notification
Amazon SNS
Status update notification is pushed to user's friends
User's friends receive status update notification

Figure 5: Mobile Backends

Record Handling utilizing AWS Lambda One common utilize case for AWS Lambda is record handling, which includes assignments such as resizing pictures, parsing log records, and extricating data from archives. Underneath is the basic case of a Lambda work that forms records when they are transferred to an S3 bucket.

4. Result and Interpretation

AWS Lambda function is the impressive feature of event-driven computing, empowers you to execute code in response to a wide range of events. Whether it's modifications to data in an Amazon S3 bucket, updates to a DynamoDB table, or even an HTTP request through Amazon API Gateway, Lambda stands ready to spring into action. With this innovative functionality at your fingertips, you can effortlessly harness the power of events to drive your code's execution.

1) Create an S3 Bucket:

To get started, login to the AWS Management Console and open the S3 service. From there, you can create a new bucket with a name that suits your needs (e.g., "my-file-processing-bucket").

2) Create a Lambda Function:

Next, open the Lambda service in the AWS Management Console. Click on "Create function" and provide a name for your function (e.g., "FileProcessingFunction"). Choose the appropriate runtime (e.g., Python 3.8) and set up the execution role with S3 read and write permissions [20].

3) Add S3 Trigger:

In the "Designer" section of the Lambda console, click on "Add Trigger" to add a trigger to your function. Choose "S3" from the list of services. Select the bucket that you created in step 1 and specify the event type that you want the function to be triggered by (e.g., "All object create events"). Click "Add" to add the trigger to your function.

4) Write File Processing Code:

Now, you can write the code for your file processing Lambda function in the "Function code" section of the console.

By following these steps, you can create a working model of an AWS Lambda function that performs file processing tasks when files are uploaded to an S3 bucket. This provides a powerful and scalable solution for automating file processing workflows in the cloud.

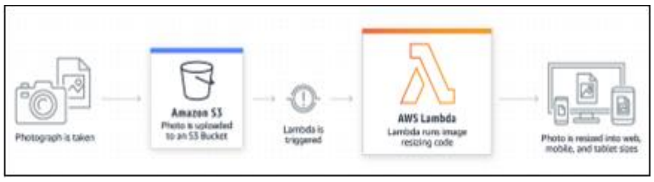
Photograph is taken
Amazon S3
Photo is uploaded to an S3 Bucket
Lambda is triggered
AWS Lambda
Lambda runs image resizing code
Photo is resized into web, mobile, and tablet sizes

Figure 6: Processing Code

Events are triggered on AWS Lambda - Events in AWS Lambda are the triggers that initiate the execution of your Lambda functions. Some of the events that can trigger AWS Lambda functions:

API Gateway – It can configure AWS Lambda to be triggered by HTTP requests through API Gateway. This is commonly used for building serverless web applications and APIs.

AWS S3 (Object Storage) -Lambda functions can be triggered when objects are created, modified, or deleted in an Amazon S3 bucket. This is useful for processing files, generating thumbnails, or performing other tasks in response to changes in S3.

AWS DynamoDB (NoSQL Database) - Lambda functions can be triggered in response to changes in DynamoDB tables. This includes events such as inserts, updates, and deletes in the database.

AWS SNS (Simple Notification Service) -Lambda functions can subscribe to SNS topics and be triggered when messages are published to those topics. This is useful for building event-driven architectures.

AWS CloudWatch Event - CloudWatch Events allow you to schedule events or define rules based on events in AWS services. Lambda functions can be triggered by these scheduled events or events that match specified rules.

AWS CloudFormation Stack Update - Lambda functions can be triggered when an AWS CloudFormation stack is created, updated, or deleted. This enables automation of tasks during the lifecycle of infrastructure.

AWS Kinesis (Data Streaming) -Lambda functions can process records from an Amazon Kinesis stream, allowing real-time processing of streaming data.

AWS Step Function - AWS Step Functions allow you to coordinate and sequence multiple AWS services, including Lambda functions. Lambda functions can be triggered as part of step.

Functions workflow- AWS IoT (Internet of Things) - Lambda functions can be triggered by messages from AWS IoT devices. This allows you to process data generated by IoT devices in real-time.

Custom Events or External Sources -You can create custom events or use external sources to trigger Lambda functions by invoking the function directly or through API Gateway.

AWS SQS (Simple Queue Service)- Lambda functions can be configured to process messages from an SQS queue. This enables building scalable and asynchronous processing systems.

5. Conclusion

The study highlights AWS Lambda's performance characteristics and offers practical optimization tools for serverless environments. It addresses the challenge of forecasting intermittent demand data, marked by numerous zeros, and introduces a modified formula for calculating the average of non-zero demands. This new approach is tested against existing methods using inventory data from Automobile Spare Parts. The results demonstrate that the modified formula significantly improves forecasting accuracy for both consistent and random demands. Overall, the proposed method enhances prediction accuracy, making it a valuable tool for handling intermittent demand data.

References

[1] L. Ruan, J. Peng, L. Xiao and X. Wang, "CloudDVMM: Distributed Virtual Machine Monitor for Cloud Computing," *2013 IEEE International Conference on Green Computing and Communications and IEEE Internet of Things and IEEE Cyber, Physical and Social Computing*, Beijing, China, 2013, pp. 1853-1858, doi: 10.1109/GreenCom-iThings-CPSCom.2013.344.

[2] P. Orduña, A. Gómez-Goiri, L. Rodriguez-Gil, J. Diego, D. López-de-Ipiña and J. Garcia-Zubia, "wCloud: Automatic generation of WebLab-Deusto deployments in the Cloud," *Proceedings of 2015 12th International Conference on Remote Engineering and Virtual Instrumentation (REV)*, Bangkok, Thailand, 2015, pp. 223-229, doi: 10.1109/REV.2015.7087296.

[3] C. -H. Hsieh *et al.*, "A Migration System of Database for Virtual Machine in Cloud Computing," *2021 4th International Conference on Information Communication and Signal Processing (ICICSP)*, Shanghai, China, 2021, pp. 582-587, doi: 10.1109/ICICSP54369.2021.9611882.

[4] J. Han, M. Song and J. Song, "A Novel Solution of Distributed Memory NoSQL Database for Cloud Computing," *2011 10th IEEE/ACIS International Conference on Computer and Information Science*, Sanya, China, 2011, pp. 351-355, doi: 10.1109/ICIS.2011.61.

[5] Aryachandra, Y. F. Arif and S. N. Anggis, "Intrusion Detection System (IDS) server placement analysis in cloud computing," *2016 4th International Conference on Information and Communication Technology (ICoICT)*, Bandung, Indonesia, 2016, pp. 1-5, doi: 10.1109/ICoICT.2016.7571954.

[6] T. Zhang and R. B. Lee, "Monitoring and Attestation of Virtual Machine Security Health in Cloud Computing," in *IEEE Micro*, vol. 36, no. 5, pp. 28-37, Sept.-Oct. 2016, doi: 10.1109/MM.2016.86. keywords: {Systems modelling; Cloud

[7] S. Garg, D. D. V. Gupta and R. K. Dwivedi, "Enhanced Active Monitoring Load Balancing algorithm for Virtual Machines in cloud computing," *2016 International Conference System Modeling & Advancement in Research Trends (SMART)*, Moradabad, India, 2016, pp. 339-344, doi: 10.1109/SYSMART.2016.7894546.

[8] P. K. Tiwari and S. Joshi, "Dynamic weighted virtual machine live migration mechanism to manages load balancing in cloud computing," *2016 IEEE International Conference on Computational Intelligence and Computing Research (ICCIC)*, Chennai, India, 2016, pp. 1-5, doi: 10.1109/ICCIC.2016.7919581.

[9] V. Kumar and R. S. Rathore, "Security Issues with Virtualization in Cloud Computing," *2018 International Conference on Advances in Computing, Communication Control and Networking (ICACCCN)*, Greater Noida, India, 2018, pp. 487-491, doi: 10.1109/ICACCCN.2018.8748405.

[10] S. Garg, R. K. Dwivedi and H. Chauhan, "Efficient utilization of virtual machines in cloud computing using Synchronized Throttled Load Balancing," *2015 1st International Conference on Next Generation Computing Technologies (NGCT)*, Dehradun, India, 2015, pp. 77-80, doi: 10.1109/NGCT.2015.7375086.

[11] C. Gong, J. Liu, Q. Zhang, H. Chen and Z. Gong, "The Characteristics of Cloud Computing," *2010 39th International Conference on Parallel Processing Workshops*, San Diego, CA, USA, 2010, pp. 275-279, doi: 10.1109/ICPPW.2010.45.

[12] P. Das and P. M. Khilar, "VFT: A virtualization and fault tolerance approach for cloud computing," *2013 IEEE Conference on Information & Communication Technologies*, Thuckalay, India, 2013, pp. 473-478, doi: 10.1109/CICT.2013.6558142.

[13] V. Tkachov, M. Hunko and V. Volotka, "Scenarios for Implementation of Nested Virtualization Technology in Task of Improving Cloud Firewall Fault Tolerance," *2019 IEEE International Scientific-Practical Conference Problems of Infocommunications, Science and Technology (PIC S&T)*, Kyiv, Ukraine, 2019, pp. 759-763, doi: 10.1109/PICST47496.2019.9061473.

[14] L. Ruan, J. Peng, L. Xiao and X. Wang, "CloudDVMM: Distributed Virtual Machine Monitor for Cloud Computing," *2013 IEEE International Conference on Green Computing and Communications and IEEE*

Internet of Things and IEEE Cyber, Physical and Social Computing, Beijing, China, 2013, pp. 1853-1858, doi: 10.1109/GreenCom-iThings-CPSCom.2013.344.

[15] Sarker, I.H., Furhad, M.H. & Nowrozy, R. AI-Driven Cybersecurity: An Overview, Security Intelligence Modeling and Research Directions. *SN COMPUT. SCI.* 2, 173 (2021). https://doi.org/10.1007/s42979-021-00557-0

[16] Duan, Qiang, et al. "Combined federated and split learning in edge computing for ubiquitous intelligence in internet of things: State-of-the-art and future directions." *Sensors* 22.16 (2022): 5983.

[17] Singh, Raghubir, and Sukhpal Singh Gill. "Edge AI: a survey." *Internet of Things and Cyber-Physical Systems* (2023).

[18] Anders Segerstedt, Erik Levén. "A Study of Different Croston-Like Forecasting Methods." Operations and Supply Chain Management An International Journal, Volume 16, Issue 3, 2023. Available from: https://doi.org/10.31387/oscm0540400.

[19] Takumi Kato. “Demand Prediction in the Automobile Industry Independent of Big Data”. Annals of Data Science. Springer.9 (2): 249-270. (2022). https://doi.org/10.1007/s40745-020-00278-w

[20] JM Rožanec, B. Fortuna, D. Mladenić. "Reframing demand forecasting: a two-fold approach for lumpy and intermittent demand." Sustainability 2022, 14(15), 9295. Available from: https://doi.org/10.3390/su14159295.

[21] Rožanec, J. M., Kažič, B., Škrjanc, M., Fortuna, B., Mladenić, D. Automotive OEM Demand Forecasting: A Comparative Study of Forecasting Algorithms and Strategies. Applied Sciences. 11(15). 6787. (2021). https://doi.org/10.3390/app11156787

[22] Çerağ Pinçe, Laura Turrini, Joern Meissner, Intermittent demand forecasting for spare parts: A Critical review, Omega,Volume 105,2021,https://doi.org/10.1016/j.omega.2021.102513.

[23] M. Hasni, M.S. Aguir, M.Z. Babai, Z. Jemai. "On the performance of adjusted bootstrapping methods for intermittent demand forecasting." International Journal of Production Economics, Volume 216, October 2019, Pages 145-153. Available from: https://doi.org/10.1016/j.ijpe.2019.04.005.

[24] TE YİLMAZ, G. YAPAR, İ. YAVUZ. "Comparison of Ata method and Croston based methods on forecasting of intermittent demand." Mugla Journal of Science and Technology, Volume 49-55, December 2019. Available from: https://doi.org/10.22531/muglajsci.572444.