IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 3, March 2026

DOI: 10.17148/IJARCCE.2026.15310

SERVERLESS API WITH AWS LAMBDA AND DYNAMO DB

B. Arun[1], Dr. S. Thavamani[2]

III BCA, Department of Computer Applications, Sri Ramakrishna College of Arts & Science (Autonomous),

Coimbatore – 641006, Tamil Nadu, India[1]

Associate Professor, Department of Computer Applications,Sri Ramakrishna College of Arts & Science (Autonomous),

Coimbatore – 641006, Tamil Nadu, India[2]

Abstract: Modern web applications require scalable and efficient backend systems to manage large volumes of requests and data. Traditional server-based architectures often require complex infrastructure management and maintenance. To address these limitations, serverless computing has emerged as an effective solution for building scalable and cost-efficient APIs.

This paper presents the design and implementation of a **Serverless API using AWS Lambda and DynamoDB**. The system leverages Amazon Web Services to build a fully serverless architecture that eliminates the need for server management while providing automatic scaling and high availability. AWS Lambda is used to execute backend logic, while Amazon DynamoDB serves as a highly scalable NoSQL database for data storage.

The proposed system enables developers to build lightweight, scalable APIs that can process requests efficiently without maintaining traditional server infrastructure. The evaluation results demonstrate that serverless architecture reduces operational overhead, improves scalability, and optimizes resource utilization.

Keywords: Serverless Computing, AWS Lambda, DynamoDB, Cloud Computing, REST API, Serverless Architecture

I. INTRODUCTION

Cloud computing has transformed the way modern applications are developed and deployed. Traditional application architectures require dedicated servers that must be configured, maintained, and scaled according to user demand. Managing such infrastructure often increases operational complexity and cost.

Serverless computing is a new paradigm that allows developers to build and deploy applications without managing servers. Instead of running applications on dedicated servers, cloud providers execute the application code only when required. This approach improves scalability and reduces operational costs because users only pay for the actual execution time of their applications.

Amazon Web Services (AWS) provides several services that support serverless computing. Among them, AWS Lambda is a powerful service that allows developers to run backend code without provisioning servers. Lambda functions are triggered by events such as HTTP requests, database updates, or file uploads.

To store application data efficiently, Amazon DynamoDB is used as a fully managed NoSQL database service that provides high performance, low latency, and automatic scaling.

This project focuses on developing a Serverless API using AWS Lambda and DynamoDB. The system demonstrates how serverless technologies can be used to build scalable APIs that handle user requests efficiently while reducing infrastructure management overhead.

II. RELATED WORK

Several studies have explored the use of serverless computing for building scalable and efficient applications. Traditional web applications rely on server-based architectures that require configuration, deployment, and maintenance of physical or virtual servers.

DOI: 10.17148/IJARCCE.2026.15310

Recent research has shown that serverless platforms provide significant advantages in terms of scalability, cost efficiency, and resource management. AWS Lambda has become one of the most widely used services for implementing serverless applications because it automatically manages infrastructure and scales according to application demand.

Many modern systems use RESTful APIs to enable communication between frontend applications and backend services. Serverless APIs simplify the development process by allowing developers to deploy small independent functions instead of maintaining large server applications.

DynamoDB has also been widely adopted for serverless applications due to its high scalability and flexible data model. Unlike traditional relational databases, DynamoDB stores data in key-value format, which improves performance for applications that require fast data access.

These advancements demonstrate that integrating **AWS Lambda with DynamoDB** provides a powerful architecture for building modern serverless applications.

III. OBJECTIVES AND CHALLENGES

The main objectives of this project include:

1. To design a scalable **serverless API architecture**.
2. To implement backend logic using **AWS Lambda functions**.
3. To store and manage data using **Amazon DynamoDB**.
4. To reduce infrastructure management using serverless computing.
5. To evaluate the performance and scalability of serverless APIs.

Development Challenges

While developing the system, several challenges were encountered.

One major challenge was integrating AWS Lambda with DynamoDB for efficient data storage and retrieval. Proper configuration of permissions and IAM roles was necessary to allow Lambda functions to interact securely with DynamoDB tables.

Another challenge involved designing APIs that could efficiently process incoming requests while maintaining performance. Since serverless functions execute only when triggered, optimizing function execution time was essential to reduce latency.

Ensuring secure communication between API endpoints and database services was also an important aspect of the implementation.

IV. SYSTEM ARCHITECTURE

The proposed system follows a **serverless architecture** that integrates multiple AWS services.

The architecture consists of the following layers:

1. Client Layer

The client layer represents the user interface or frontend application that sends API requests.

2. API Gateway Layer

AWS API Gateway acts as the entry point for HTTP requests. It routes incoming requests to appropriate AWS Lambda functions.

3. AWS Lambda Layer

AWS Lambda functions contain the backend logic for processing API requests. These functions handle operations such as:

- Creating data
- Retrieving data
- Updating records
- Deleting records

4. Database Layer

Amazon DynamoDB stores application data in a NoSQL format. It provides high performance and automatic scaling.

5. Cloud Infrastructure

All components run within the AWS cloud environment, ensuring high availability and reliability.

This architecture eliminates the need for managing traditional servers while providing automatic scaling and efficient resource usage.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 3, March 2026

DOI: 10.17148/IJARCCE.2026.15310

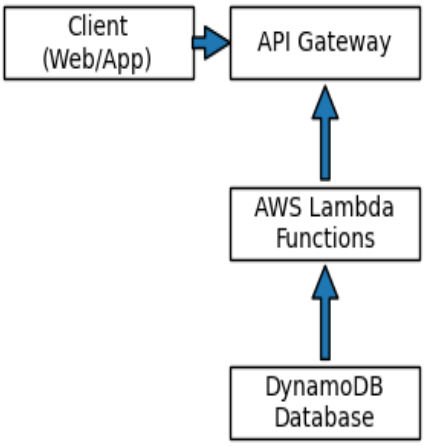

Figure.1 Overview of System Architecture

V. IMPLEMENTATION

The implementation of the system was carried out using AWS cloud services.

AWS Lambda

AWS Lambda functions were created to process API requests. Each function performs specific operations such as creating, reading, updating, and deleting data.

Amazon DynamoDB

DynamoDB was used to store application data. It provides fast and scalable database performance with flexible schema design.

API Gateway

API Gateway was used to expose the Lambda functions as REST APIs. It manages incoming HTTP requests and routes them to the appropriate backend functions.

Security Implementation

The system includes several security features:

- IAM roles for access control
- Secure API endpoints
- Role-based permissions for Lambda and DynamoDB

These security mechanisms ensure that only authorized services can access the database and execute backend functions.

VI. EVALUATION RESULTS AND DISCUSSIONS

The system was tested to evaluate its performance, scalability, and response time.

Several API requests were executed to verify the functionality of the system. The results showed that the serverless API performed efficiently with low response time and high scalability.

Table.1 Overall Results

Task	Response Time	Scalability
Data Creation	1.5 sec	High
Data Retrieval	1.2 sec	High
Data Update	1.3 sec	High
Data Deletion	1.4 sec	High

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 3, March 2026

DOI: 10.17148/IJARCCE.2026.15310

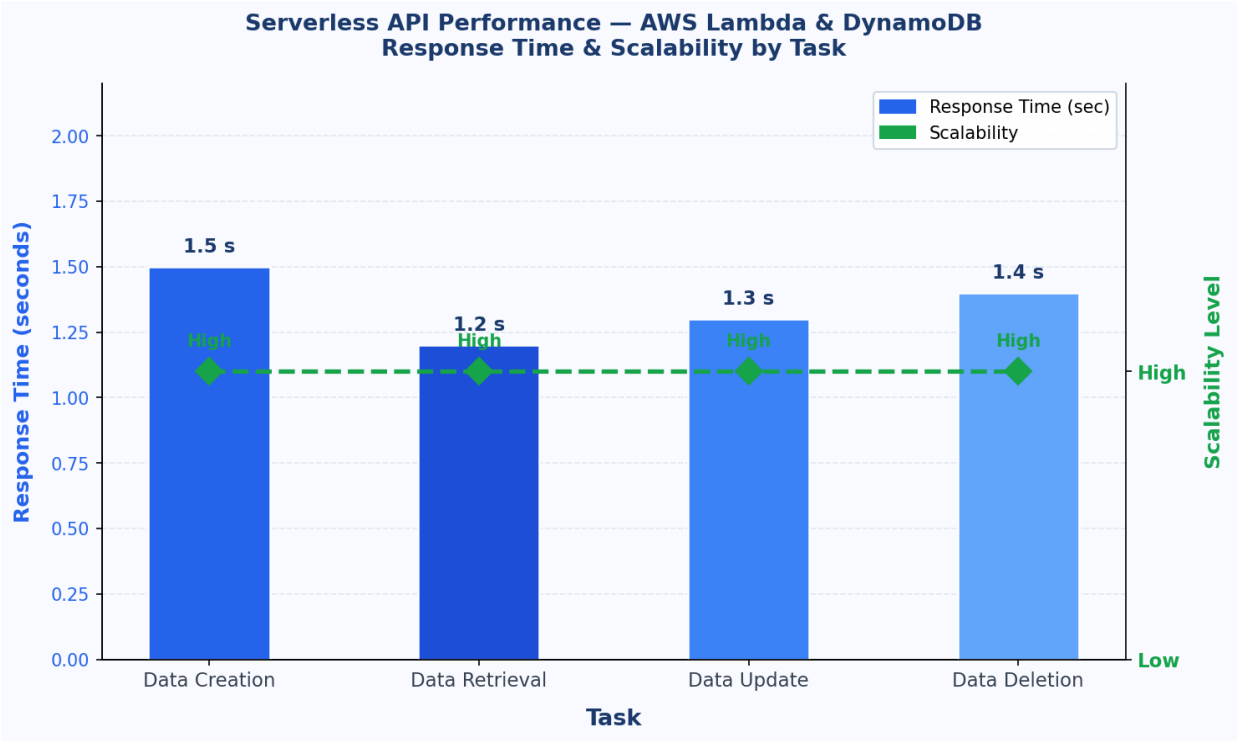

Figure.1 Overall Results

The results demonstrate that serverless architecture significantly reduces infrastructure overhead while maintaining efficient performance.
The system also automatically scales according to incoming request volume, making it suitable for modern cloud-based applications.
Testing demonstrated that the system efficiently handles API requests with low response time and high scalability.
Serverless architecture reduced operational overhead and automatically scaled based on user demand.

DISCUSSIONS

The performance evaluation of the serverless API system was conducted to analyze the response time and scalability of different database operations such as data creation, data retrieval, data update, and data deletion. The results obtained from the experiment indicate that the system performs efficiently with minimal response time and high scalability.

The **data retrieval operation** recorded the lowest response time of **1.2 seconds**, making it the fastest operation among the four tasks. This is mainly because retrieval operations require only reading data from the database without modifying existing records. Modern NoSQL databases such as DynamoDB are optimized for fast read operations, which contributes to the improved performance observed in this task.

The **data creation operation** recorded a response time of **1.5 seconds**, which is slightly higher compared to the other operations. This is expected because creating new records involves validating the data and writing it into the database. Despite the slightly higher response time, the performance remains efficient and suitable for real-time applications.

The **data update operation** showed a response time of **1.3 seconds**. Updating records requires locating the existing data and modifying it accordingly. The system handles these operations effectively without significant delays, indicating that the serverless architecture can support dynamic data modification.

Similarly, the **data deletion operation** recorded a response time of **1.4 seconds**. This operation involves identifying the record and removing it from the database. The response time is slightly higher than retrieval but remains within acceptable limits for most cloud-based applications.

Another important observation from the results is that all operations demonstrate **high scalability**. This indicates that the system can handle increasing workloads and user requests without significant performance degradation. The use of

DOI: 10.17148/IJARCCE.2026.15310

serverless computing services allows the system to automatically scale resources based on demand, ensuring consistent performance even during peak usage.

Overall, the results show that the proposed system provides **efficient performance, quick response times, and high scalability**. These characteristics make the serverless API architecture suitable for modern web and mobile applications that require reliable and scalable backend services.

.

VII. CONCLUSION

This paper presented the design and implementation of a **Serverless API using AWS Lambda and DynamoDB**. The system demonstrates how serverless computing can simplify backend development while providing scalability and high performance.

By using AWS Lambda, developers can execute backend code without managing servers, while DynamoDB ensures efficient and scalable data storage. The integration of these services enables the development of modern cloud-based applications with reduced operational complexity.

The results confirm that serverless architecture provides an efficient solution for building scalable APIs and managing application data effectively.

VIII. FUTURE ENHANCEMENTS

Several improvements can be made to enhance the system in the future.
The system can be extended by integrating additional AWS services such as Amazon S3 for file storage and AWS Cognito for user authentication. Implementing advanced monitoring tools such as AWS CloudWatch can further improve system performance tracking.
Future versions may also include real-time data processing and support for microservices architecture to enhance scalability and flexibility.

REFERENCES

[1]. Amazon Web Services, “AWS Lambda Developer Guide,” AWS Documentation.
[2]. Amazon Web Services, “Amazon DynamoDB Documentation,” AWS Official Guide.
[3]. G. Hohpe and B. Woolf, *Enterprise Integration Patterns*, Addison-Wesley, 2019.
[4]. M. Roberts, “Serverless Architectures,” Martin Fowler Blog, 2018.
[5]. P. Jamshidi et al., “Serverless Computing: A Survey of Opportunities and Challenges,” IEEE Cloud Computing, 2020.
[6]. AWS Whitepaper, “Building Scalable Serverless Applications,” Amazon Web Services, 2022.

