A R DIGITECH
International Journal Of Engineering, Education And Technology (ARDIJEET)
www.ardigitech.in ISSN 2320-883X, VOLUME 13 ISSUE 01, 01/01/2025 pub: 26/02/2025

Systematic Research Using Serverless Functions For Cloud-Native Application

[1]Ashish Narkhede, [2]Prof. Pallavi P. Rane, [3]Prof. Pravin S. Rane, [4]Prof. Nilesh N. Shingne
[1]ashishnarkhede100@gmail.com, [2]koltepallavi200@gmail.com, [3]pravin28.rane@gmail.com
[4]shingne.nilesh236@gmail.com
[2,3]Assistant professor, Rajarshi Shahu College of Engineering, Buldhana, Maharashtra
[4]Assistant professor, Sanmati engineering College, Washim, Maharashtra

Abstract: Cloud-native computing has fundamentally changed how applications are built, offering solutions that are scalable, cost-effective, and adaptable. This research explores the creation and execution of a cloud-native application using serverless functions, a design approach that removes the requirement for server management while maximizing resource usage. The application uses Function-as-a-Service (FaaS) platforms to dynamically run separate, event-driven functions in response to user and system activities. Important elements include API gateways for smooth client interaction, event triggers for real-time processing, and scalable databases for effective data storage. The suggested design highlights modularity, allowing for easier upgrades and maintenance, and assures high availability by utilizing distributed cloud infrastructure. Security measures, such as identity and access management (IAM) and encrypted data transmission, are included to secure the application. The implementation is validated by performance benchmarking, which shows lower latency and cost-effectiveness when compared to traditional monolithic or microservices designs. This research illustrates the potential of serverless computing for speeding up development cycles and supporting modern, resilient cloud-native applications in a variety of industries. [3]

Keywords: Cloud, Scalable, Distributed, Identity, Computing

1. Introduction

Function-as-a-Service (FaaS), or serverless functions, is a cloud computing model where developers create and execute applications without managing the underlying server infrastructure. These functions are triggered by specific events, like HTTP requests, file uploads, or database changes. Building serverless functions requires a different approach than traditional application development. They should be small, focused, and stateless, each with a single, well-defined purpose. Each function should handle one specific task. This promotes modularity, reusability, and simplified maintenance. Functions shouldn't store data between executions. Any required data persistence should be handled by external services like databases or object storage. Clearly defined input and output structures are essential. This ensures smooth communication and integration between different functions and services. Functions should produce the same result regardless of how many times they're executed with the same input. This improves data consistency and simplifies retry mechanisms. Setting appropriate timeouts and resource limits prevents runaway functions from impacting overall system performance.

2. Project Objectives:

- To design applications that automatically scale based on demand without manual intervention or resource provisioning.
- To optimize resource usage by executing code only when necessary, paying only for the actual compute time used.

- To accelerate development by allowing developers to focus on writing code rather than managing infrastructure.

3. Literature Review

The advent of cloud computing has caused a sea shift in the app development life cycle. Cloud computing has come a long way, with traditional models like Infrastructure-as-a-Service (Iaas) and Platform-as-a-Service (Pass) allowing businesses to tap into computer resources whenever they need them [1].

A new server-less computing concept has arisen to solve these problems by removing the need to maintain the underlying infrastructure [2]. Developers develop and deploy individual functions in a server-less architecture. Events or requests trigger these functions. In response to changes in demand, the cloud service provider automatically scales up or down the resources used to carry out these tasks. Without worrying about server provisioning, scalability, or maintenance, developers can concentrate entirely on building code and providing business value by utilizing this strategy. [2]

Data management in server-less systems requires a distinct strategy compared to conventional designs. Functions that do not rely on servers cannot maintain permanent connections to storage services or databases because they are stateless and have a limited execution period. Instead, information needs to be kept in third-party services designed to handle server-less access patterns. [3].

4. Motivation

Serverless functions eliminate the need for developers to manage servers, reducing the operational burden associated with infrastructure provisioning, scaling, and maintenance. This allows developers to focus solely on writing business logic, streamlining the development process. Serverless computing operates on a pay-as-you-go model, where you only pay for the resources used during function execution. This makes it a cost-effective solution for handling variable workloads. There is no need to maintain idle servers, which further optimizes costs by scaling resources dynamically. Serverless functions can automatically scale based on demand. This means they can easily handle both low and high volumes of traffic without requiring manual intervention or reconfiguration. The cloud provider takes care of scaling resources to meet the application's needs, making it ideal for fluctuating workloads. Serverless architecture accelerates application development by simplifying infrastructure management. Developers can deploy code faster, experiment with different components, and make changes without worrying about the underlying infrastructure. This results in quicker prototyping and faster product launches. Serverless design allows developers to focus purely on business logic, without getting bogged down by concerns such as infrastructure setup, capacity planning, or server maintenance. This ensures more innovation and agility in developing applications. [6]

5. Proposed Methodology:

Serverless functions operate within an event-driven architecture, where cloud providers manage infrastructure, allowing developers to focus solely on writing business logic. These functions are lightweight, short-lived, and triggered by specific events. Below is an explanation of how serverless functions operate:

- **Function Deployment:**
  Developers write small functions that handle tasks such as data processing, interacting with databases, or calling APIs. These functions are uploaded to a serverless platform (e.g., AWS Lambda, Azure Functions, or Google Cloud Functions), where they are stored and ready to be executed.
- **Event Triggering:**
  Serverless functions are activated by events. These events can include:
    - An HTTP request (e.g., a REST API call)

- A change in cloud storage (e.g., uploading a file to a storage bucket)
- A message added to a queue (e.g., a new task in a job queue)
- A scheduled task (e.g., running a function at a set time or interval)

The cloud provider monitors these events and triggers the corresponding serverless function when an event occurs.

- **Function Execution:**

  Once an event is detected, the cloud provider automatically allocates resources to execute the function. This allocation occurs on-demand, meaning no resources are used unless the function is triggered. The function runs its logic and may interact with cloud services like databases, storage, or external APIs. Serverless functions are stateless, meaning they don't retain any data or context between invocations. If the function needs to store persistent data (e.g., user sessions or database records), external services like databases or object storage are used. [7]

- **Scaling:**

  Serverless functions automatically scale to accommodate the number of incoming requests or events. If there is an increase in traffic or events, new instances of the function are created to handle the load. If the traffic decreases, the cloud provider scales down the instances, possibly terminating them if no further events are triggered. This scaling process is elastic and requires no manual intervention.

- **Execution Time and Resource Management:**

  Serverless functions are designed to execute quickly, typically within milliseconds or seconds. Platforms impose execution time limits (e.g., AWS Lambda limits execution to 15 minutes). If the function runs too long, it is terminated. Developers can define resource limits such as memory allocation and maximum execution duration to ensure efficient use of resources.

- **Cold Start:**

  When a serverless function is invoked for the first time or after a period of inactivity, it may experience a cold start. During a cold start, the platform needs to initialize resources and load the function's runtime environment, which can cause slight delays. Once the function has run once, it remains "warm" for subsequent invocations, reducing the latency caused by cold starts.

- **Response and Termination:**

  After execution, the function returns a response, such as JSON, HTML, or other data formats, based on the task it performed. Once the function completes its job, it terminates, and the cloud provider automatically releases the allocated resources. The platform handles the management and cleanup of these resources. [8]

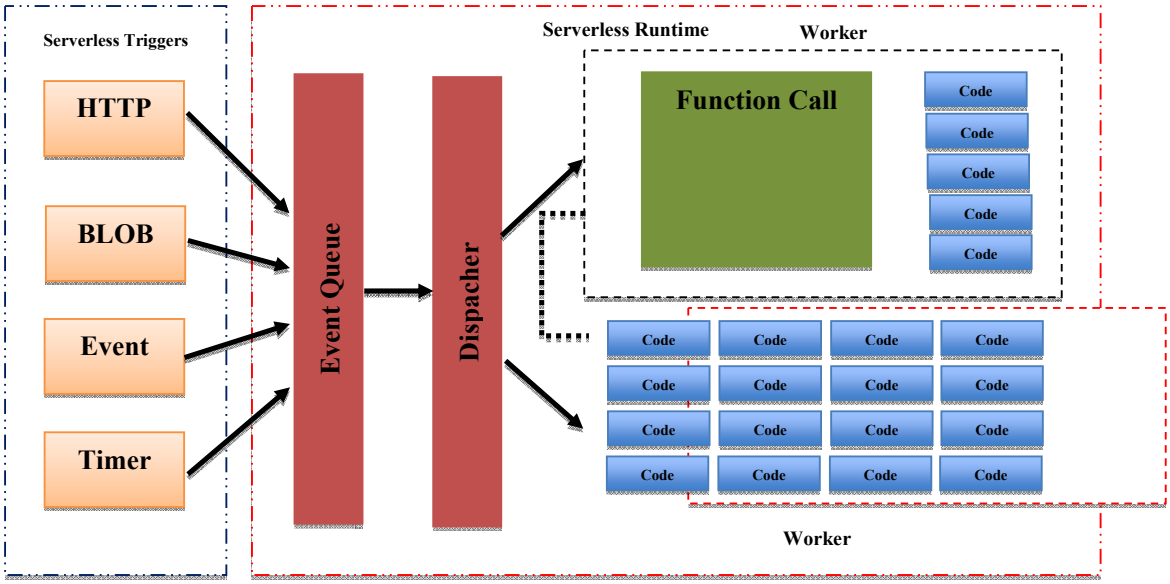

Fig. 1 Flow of the proposed system

6. Expected Outcomes

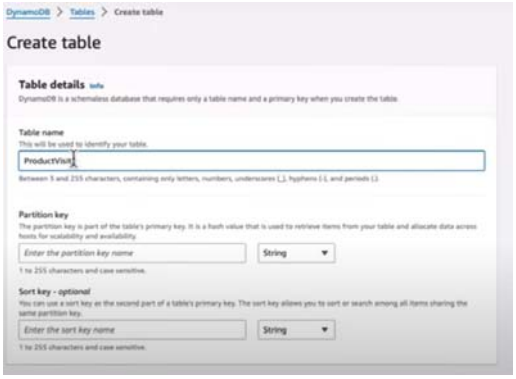

Fig. 2 Creating database

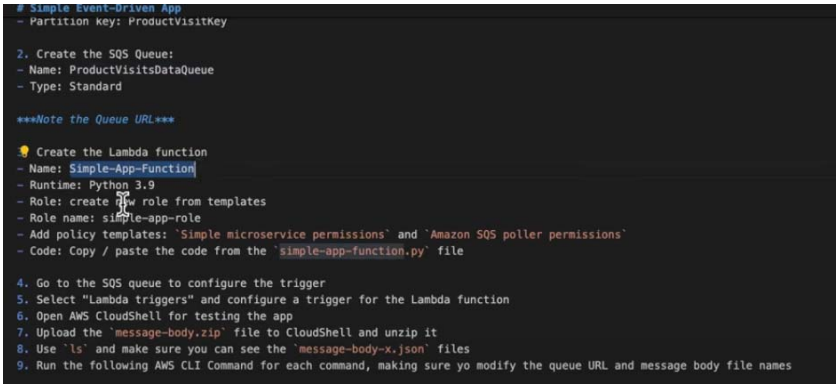

Fig. 3 Configuring an app

Fig. 4 Storing of file in application

7. Benefits

Serverless functions offer numerous advantages, making them a compelling choice for cloud-native applications and microservice architectures. These benefits include:

- **Cost-Effectiveness:** A pay-as-you-go model means you only pay for the compute time used, not idle time. This significantly reduces costs, especially for workloads with fluctuating or unpredictable usage.
- **Automatic Scaling:** Serverless functions automatically scale based on demand. The cloud provider manages resource allocation, ensuring efficient use without manual intervention.
- **Simplified Infrastructure:** Developers don't manage servers. This allows them to focus on code and application features, not infrastructure provisioning, maintenance, or scaling.
- **Faster Deployment:** Serverless platforms accelerate development and deployment. Code changes can be deployed quickly without infrastructure configuration, enabling rapid updates, testing, and iteration.
- **High Availability:** Built-in fault tolerance and deployment across multiple availability zones ensure minimal downtime. If one function instance fails, another takes over.
- **Event-Driven:** Triggered by events (HTTP requests, database changes, etc.), serverless functions are ideal for event-driven applications like real-time processing, automation, and microservice communication.
- **Reduced Operational Complexity:** No server, load balancer, or database management simplifies operations, allowing teams to focus on coding and innovation.
- **Enhanced Security:** Serverless platforms often include built-in security features like automatic patching, network isolation, and data encryption, minimizing security concerns.
- **Flexibility and Agility:** Working with individual components enables modular application development, facilitating rapid changes and adaptations to evolving business needs.
- **Efficient Resource Use:** Automatic scaling ensures resources are only used when needed, maximizing efficiency and minimizing waste compared to traditional systems.
- **Global Reach:** Global infrastructure allows functions to run in multiple regions, improving performance and providing localized services.
- **Focus on Business Logic:** Developers can concentrate on code and business logic, while the cloud provider handles infrastructure, scaling, and resources, enabling faster innovation and better user experiences.

8. Conclusion:

This paper examines the implementation of serverless computing for building scalable and cost-effective cloud applications. It covers the fundamental concepts, benefits, and challenges of serverless computing, and proposes a framework for designing and deploying such architectures. Performance, scalability, and cost-effectiveness are evaluated across different cloud platforms and workloads through experimentation. The results demonstrate the significant advantages of serverless architectures, including lower operational overhead, improved scalability, and cost reductions. The paper also discusses best practices and identifies future research directions for addressing current limitations and optimizing serverless adoption. Expected advancements in orchestration, state management, performance, and security will further enhance serverless computing. In summary, serverless computing offers a compelling approach for developing scalable and cost-efficient cloud applications. By leveraging the benefits of serverless architectures and following best practices, organizations can improve agility, efficiency, and innovation in their cloud-native application development and deployment.

References:

[1] Maciej Malawski et al., "Server Less Execution of Scientific Workflows: Experiments with Hyper Flow, AWS Lambda and Google Cloud Functions," Future Generation Computer Systems, vol. 110, pp. 502-514, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[2] Ioana Baldini et al., "Server Less Computing: Current Trends and Open Problems," Research Advances in Cloud Computing, pp. 1-20, 2017. [CrossRef] [Google Scholar] [Publisher Link]

[3] CNCF Survey 2020, Cloud Native Computing Foundation, 2020. [Online]. Available: https://www.cncf.io/wp-content/uploads/2020/11/CNCF_Survey_Report_2020.pdf

[4] J. Schiller-Smith et al., "The Server Less Dilemma: Function Composition for Server Less Computing," Proceedings of the ACM Symposium on Cloud Computing, pp. 347-362, 2019.

[5] Paul Castro et al., "The Rise of Server Less Computing," Communications of the ACM, vol. 62, no. 12**,** pp. 44-54, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[6] Garrett McGrath, and Paul R. Brenner, "Server Less Computing: Design, Implementation, and Performance," IEEE 37th International Conference on Distributed Computing Systems Workshops, Atlanta, GA, USA, pp. 405-410, 2017. [CrossRef] [Google Scholar] [Publisher Link]

[7] Mohit Sewak, and Sachchidanand Singh, "Winning in the Era of Server Less Computing and Function as a Service," 3rd International Conference on Computing for Sustainable Global Development, Pune, India, pp. 1169-1175, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[8] Johannes Manner et al., "Cold Start Influencing Factors in Function as a Service," IEEE/ACM International Conference on Utility and Cloud Computing Companion, Zurich, Switzerland, pp. 181-188, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[9] Adam Eivy, and Joe Weinman, "Be Wary of the Economics of "Server Less" Cloud Computing," IEEE Cloud Computing, vol. 4, no. 2, pp. 6-12, 2017. [CrossRef] [Google Scholar] [Publisher Link]

[10] Vipul Gupta et al., "Over Sketch: Approximate Matrix Multiplication for the Cloud," IEEE International Conference on Big Data, Seattle, WA, USA, pp. 298-304, 2018. [CrossRef] [Google Scholar] [Publisher Link]