MANTECH Publications
Recent Trends in Computer Science and Software Technology
Volume 5, Issue 2, May-August 2020

Serverless Computing: Revolutionizing Cloud-Native Development Paradigms

Dr. Ananya Kapoor
Associate Professor
Department of Computer Science
New Dawn Institute of Technology, Pune, India
Email: *ananya.kapoor@ndit.ac.in*

Mr. Ritesh Malhotra
Assistant Professor
School of Computing
Rajiv Gandhi Technical University, Bhopal, India
Email: *ritesh.malhotra@rgtu.edu.in*

Abstract

Serverless computing has emerged as a transformative approach within the cloud-native ecosystem by abstracting away server management and infrastructure provisioning. This paradigm empowers developers to focus purely on code while dynamically allocating compute resources based on execution needs. Unlike traditional cloud services, serverless models such as Function-as-a-Service (FaaS) provide event-driven execution, automatic scaling, and cost efficiency. This paper presents an in-depth review of serverless computing, its architecture, key providers, benefits, challenges, and use cases. Further, it explores how this model influences software development lifecycles, DevOps practices, and real-world enterprise adoption. The ethical and operational implications are also discussed, offering a balanced perspective on this emerging trend in cloud-native computing.

Page 49-54 © MANTECH PUBLICATIONS 2020. All Rights Reserved

Keywords: *Serverless Computing, Function-as-a-Service (FaaS), Cloud-Native, Event-Driven Architecture, DevOps, AWS Lambda, Edge Computing*

INTRODUCTION

The evolution of cloud computing has dramatically changed the way applications are built and deployed. One of the most disruptive developments in this space is **serverless computing**. Contrary to its name, serverless does not mean the absence of servers, but rather the abstraction of server management from developers. It allows developers to run code in response to events without provisioning or managing servers, using platforms such as **AWS Lambda, Azure Functions**, or **Google Cloud Functions**.

Serverless computing is an implementation of Function-as-a-Service (FaaS), which introduces a new level of flexibility and cost-effectiveness. This paper explores the core concepts, technical architecture, advantages, and limitations of serverless computing, while also considering real-world implications for developers and businesses.

ARCHITECTURE OF SERVERLESS COMPUTING

Serverless architecture comprises three major components:

- **FaaS layer**: Manages execution of short-lived stateless functions.
- **API Gateway**: Acts as a mediator between the user and serverless function.
- **Event Sources**: Triggers execution based on events (HTTP calls, DB changes, etc.).

Below is a comparison of traditional and serverless models:

Table 1: Traditional vs Serverless Architecture

Architecture Model	Management Required	Billing Model	Scalability	Suitable For
Traditional Cloud	Manual server setup	Fixed/hourly	Manual	Stateful apps
Serverless	Fully abstracted	Per execution	Automatic	Event-driven apps

Page 49-54 © MANTECH PUBLICATIONS 2020. All Rights Reserved

3. KEY CHARACTERISTICS AND BENEFITS

Event-Driven Execution

Functions are invoked automatically in response to events such as database changes, HTTP requests, or message queue events. This enables highly decoupled systems.

Scalability and Efficiency

Serverless platforms automatically scale functions in real time. This elasticity ensures optimal performance during traffic spikes and minimizes idle time.

Cost Optimization

Billing is based on actual usage (i.e., time functions run), offering significant cost advantages over traditional cloud models.

Focus on Business Logic

Developers can write code without worrying about server provisioning or scaling, leading to faster development cycles.

4. SERVERLESS PLATFORMS AND PROVIDERS

Several major cloud providers offer robust serverless platforms:

Table 2: Major Serverless Platforms

Provider	Service Name	Language Support	Unique Feature
AWS	Lambda	Python, Node.js, Java, Go	Integrated with 200+ AWS services
Microsoft Azure	Azure Functions	C#, JavaScript, Python	Strong IDE and DevOps integration
Google Cloud	Cloud Functions	Go, Node.js, Python	Tight integration with Firebase
IBM	Cloud Functions	Swift, JavaScript	Built on Apache OpenWhisk

Page 49-54 © MANTECH PUBLICATIONS 2020. All Rights Reserved

USE CASES OF SERVERLESS COMPUTING

Real-Time Data Processing

Serverless functions are ideal for log analysis, data transformation, or processing streams from IoT devices.

Chatbots and APIs

FaaS enables building responsive APIs and chatbot backends that scale on demand.

Scheduled Tasks and Automation

Serverless functions can be scheduled (cron jobs) for automated tasks like backups, reports, or cleanup processes.

CHALLENGES AND LIMITATIONS

Cold Starts

Initial invocations can have latency due to container spin-up time, which may affect performance-critical applications.

Debugging and Monitoring

Lack of traditional server visibility complicates debugging and monitoring function executions.

Vendor Lock-In

Platform-specific implementation can make it difficult to migrate between providers.

Stateless Nature

Serverless functions cannot maintain internal state between invocations. External storage or databases are required.

INFLUENCE ON SOFTWARE DEVELOPMENT

The shift toward serverless impacts various software engineering practices:

Page 49-54 © MANTECH PUBLICATIONS 2020. All Rights Reserved

- **DevOps and CI/CD**: Integration with deployment pipelines allows continuous delivery of functions.
- **Microservices Architecture**: Each function acts as a microservice, enabling better modularity.
- **Developer Productivity**: Faster iteration cycles due to infrastructure abstraction.

8. SERVERLESS VS CONTAINERIZATION

Serverless and containerized deployments (e.g., using Docker, Kubernetes) are often compared.

Table 3: Serverless vs Container-Based Deployment

Feature	**Serverless**	**Containers**
Start Time	Milliseconds (cold start)	Seconds
Management	No management	Requires orchestration (K8s)
Ideal Use Case	Short, event-driven tasks	Long-running, stateful apps
Cost Model	Per request	Per resource allocation

FUTURE OUTLOOK AND TRENDS

Serverless computing is evolving rapidly with trends like:

- **Edge Computing Integration**: Running functions closer to the data source.
- **Serverless Databases**: Combining compute and storage abstraction.
- **Hybrid and Multi-Cloud Serverless**: Avoiding vendor lock-in through abstraction layers.
- **Security Enhancements**: Built-in identity and access management.

CONCLUSION

Serverless computing signifies a paradigm shift in cloud-native development. It simplifies deployment, enhances scalability, and reduces costs. However, it also introduces new complexities in debugging, vendor dependency, and state management. For startups and enterprises alike, serverless offers a powerful toolset to innovate rapidly while maintaining

Page 49-54 © MANTECH PUBLICATIONS 2020. All Rights Reserved

operational efficiency. With continued advancement and better tooling, serverless computing is poised to become a cornerstone of modern software architectures.

REFERENCES

1. Gojko Adzic and Robert Chatley, "Serverless Computing: Economic and Architectural Impact," in ACM Software Engineering Notes, vol. 44, no. 4, July 2019.
2. Jonas Bonér, "Serverless and Stateful Functions: The Next Evolution of Microservices," IEEE Software, vol. 36, no. 1, 2019, pp. 82–89.
3. Amazon Web Services, "AWS Lambda Documentation." [Online]. Available: https://docs.aws.amazon.com/lambda/
4. Google Cloud, "Cloud Functions Overview." [Online]. Available: https://cloud.google.com/functions
5. Brendan Burns, "The Future of Serverless Computing," Communications of the ACM, vol. 63, no. 12, 2020.
6. Microsoft Azure, "Azure Functions." [Online]. Available: https://azure.microsoft.com/en-in/services/functions/
7. K. Mugan, "Serverless Computing Demystified," IEEE Cloud Computing, vol. 5, no. 3, 2018.
8. IBM Cloud, "IBM Cloud Functions." [Online]. Available: https://www.ibm.com/cloud/functions

Page 49-54 © MANTECH PUBLICATIONS 2020. All Rights Reserved