Serverless Computing (AWS): Benefits and Challenges in Modern Cloud Architecture

Miss Pooja Pradeep Kharade
Student
Master of Science in Information Technology (M.Sc. I.T.) – Part I
Bunts Sangha's S. M. Shetty College of Science, Commerce and Management Studies Powai, Mumbai.

Abstract**—** Serverless computing, particularly through AWS Lambda, offers a powerful way to build scalable applications without the need to manage servers. This paper explores the key benefits, such as automatic scaling and cost efficiency, while also addressing the challenges, including cold start latency, execution time limits, and vendor lock-in. We propose practical solutions to reduce these limitations, such as using warm instances for latency reduction, breaking down long tasks into smaller functions, and adopting multi-cloud strategies to avoid reliance on a single provider. Our proposed solutions aim to make serverless computing more efficient and flexible for businesses in today's cloud environments.

Keywords— serverless, cloud computing, infrastructure management, scalability.

I. Introduction

Cloud computing emerged after the appearance of virtualization in software and hardware infrastructures; hence cloud providers increasingly adopted it to offer their services to customers [1, 2]. Customers can access these cloud services via the Internet. Software developers have been using cloud technologies in their software solutions owing to their benefits including scalability, availability, and flexibility [3]. In general, cloud computing is divided into three main categories based on the provision of services, which are software as a service (SaaS), platform as a service (PaaS), and infrastructure as a service (IaaS). In the SaaS category, cloud providers offer different types of software as services to the users. For example, Google provides many applications as a service (e.g., Gmail, Google docs, Google sheets, and Google forms). In this type of cloud, the user is not responsible for the services development, deployment, and management. The user here only uses them without worrying about their settings, configurations, etc.

Meanwhile, in the PaaS, cloud companies provide services such as network access, storage, servers, and operating systems to be purchased by developers. The developers access these services to deploy, run, and manage their applications. In this kind of cloud, the developer is responsible for the deployment and management (settings and configurations) of their software to ensure that the application is running, while they do not control the services. Finally, in the IaaS category, the cloud consumers control and manage services such as network access, servers, operating systems, and storage. Managing cloud services is not an easy task at all.

II. Related Work

Serverless computing has gained significant attention over the past few years, and many studies have explored its benefits and limitations. One notable study by Jonas et al. (2019) analyzed the performance of AWS Lambda compared to traditional server-based architectures. They found that serverless computing reduces operational complexity and costs for small-scale, event-driven applications. Additionally, their research highlighted how AWS Lambda can scale automatically and efficiently handle workloads that fluctuate, making it ideal for applications with unpredictable traffic. However, the study also pointed out challenges like cold start delays and the limited execution time, which can impact performance for specific use cases such as real-time systems. Another important piece of work by Baldini et al. (2017) focuses on serverless computing's potential for building microservices. They showed how serverless platforms simplify the deployment of microservices by automatically managing infrastructure, leading to quicker development cycles. emphasize the need for better tools and frameworks to improve the transparency and management of serverless applications, especially as adoption increases across various industries.

III. Problem Statement and Data

In this section, we discuss the benefits and challenges in Modern Cloud Architecture of serverless computing in AWS.

A. Problem Statement

Serverless computing offers many advantages, but it also brings new challenges that need to be addressed to ensure its effective use in cloud architecture. The key questions we aim to explore in this paper are:

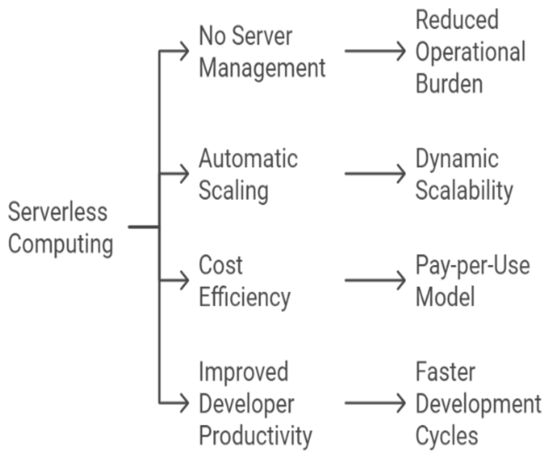
1. How can serverless computing be optimized to reduce cold start latency, especially for applications that require real-time responses?
2. What are the best practices to manage long-running tasks and applications that exceed the execution time limits imposed by platforms like AWS Lambda?
3. How can businesses avoid vendor lock-in when adopting serverless computing and ensure flexibility if they want to switch cloud providers in the future?

These questions highlight the main challenges that organizations face when using serverless computing in modern cloud architecture.

B. *Data*

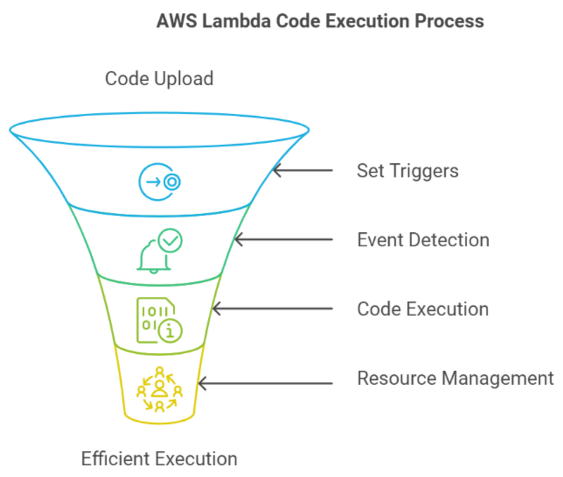
To study the benefits and challenges of serverless computing, this research focuses on data collected from various case studies and experiments using AWS Lambda. Performance metrics such as response time, scalability, and cost efficiency are examined. Additionally, the research includes examples of serverless applications from different industries, including web services and real-time data processing. This data helps analyze how serverless computing performs under different conditions and how it compares to traditional server-based models.

II. BENEFITS OF SERVERLESS COMPUTING



1. No Server Management

One of the most attractive aspects of serverless computing is the elimination of server management tasks. In traditional cloud or on-premise setups, developers need to configure, provision, and maintain the servers where their applications run. This includes tasks like installing updates, applying security patches, and scaling infrastructure during peak times. With **AWS Lambda -**



and other serverless platforms, all of this is managed automatically by the cloud provider. AWS takes care of all the infrastructure, allowing developers to focus on writing code and developing features instead of worrying about the underlying hardware. This significantly reduces the complexity and time spent on operations, which in turn speeds up development cycles. It also frees up operational teams from having to monitor and manage servers, leading to lower maintenance costs.

Use Case: A startup building a simple mobile app backend can use AWS Lambda to handle API requests without worrying about maintaining servers. Every time a user interacts with the app, AWS Lambda executes the necessary functions (like fetching or updating data) without the startup needing to manage the infrastructure. This allows the team to focus on improving the app's user experience rather than maintaining a server stack.

2. Automatic Scaling

Serverless computing offers automatic scaling, which means the system adjusts its resources dynamically based on the current demand. For example, **AWS Lambda** automatically provisions more instances of your code when there is a spike in traffic and scales back down during low activity periods. This flexibility is especially beneficial for applications with unpredictable workloads, as it ensures they can handle sudden traffic spikes without performance degradation. Unlike traditional architectures where administrators must estimate the required server capacity in advance, serverless computing removes this guesswork. AWS handles scaling automatically, ensuring that resources are only allocated as needed. This results in significant cost savings, especially for applications that experience bursts of traffic followed by inactivity.

Use Case: Consider an e-commerce website that receives a surge of traffic during holiday sales or flash sales. In such scenarios, AWS Lambda can automatically scale to handle thousands of requests simultaneously. Once the sale period ends and traffic drops, AWS Lambda scales back, and the business only pays for the compute time used during the busy period.

3. Cost Efficiency

Serverless computing is built on a **"pay-per-use"** model, meaning you are only charged for the time your code is actively running. This is in contrast to traditional servers, where you pay for the server's uptime, regardless of whether it is being fully utilized or sitting idle. For many businesses, this leads to significant cost savings, particularly for applications with sporadic or variable workloads. For example, if your application only processes requests during certain hours or under specific conditions, you won't be charged when it's inactive. This model is highly efficient for event-driven applications and workloads with unpredictable traffic, as the cost scales directly with usage. Serverless is particularly attractive for startups and businesses looking to manage their cloud budgets effectively.

Use Case: A data processing company might only need to process data files once every few hours. Instead of maintaining a server that runs 24/7, AWS Lambda can be triggered by file uploads to S3, processing the files only when needed. The company only pays for the compute time required to process each file, resulting in substantial cost savings over time.

4. Improved Developer Productivity

Serverless platforms allow developers to focus entirely on writing application logic, rather than managing infrastructure. Traditionally, developers need to work closely with DevOps teams to configure the servers, manage environment variables, and monitor the health of their applications. With serverless, much of this operational complexity is abstracted away by the cloud provider, allowing developers to work more efficiently. AWS Lambda supports a wide range of programming languages and integrates with other AWS services, making it easier to build complex workflows and applications without worrying about the underlying architecture. Developers can set up event-driven functions that automatically trigger in response to specific actions, such as database updates, file uploads, or HTTP requests. This allows for faster development cycles, which is especially beneficial for teams following agile methodologies that emphasize quick iterations.

IV. CHALLENGES OF SERVERLESS COMPUTING

Security Concerns
Cold Start Latency
Monitoring and Debugging
Limited Execution Time
Vendor Lock-In

1. Cold Start Latency

One of the most commonly reported issues with AWS Lambda and other serverless platforms is **cold start latency**. When a function is invoked after a period of inactivity (i.e., when it hasn't been used for some time), AWS needs to allocate the necessary resources, such as memory and compute, and start the container that runs the function. This process can take a few extra milliseconds to seconds, leading to a delay before the function starts executing. This delay is known as a "cold start." Although AWS has made improvements in recent years to reduce cold start times, it can still be a problem for applications that require instant response times. For instance, applications like real-time financial trading systems or time-sensitive web applications (like gaming platforms) may experience noticeable delays, which can impact user experience.

Use Case: An online store using serverless computing to process transactions might experience slight delays during periods of low activity when new requests trigger cold starts. While these delays might be negligible for most users, it could impact businesses that need faster response times, especially in high-stakes environments like payment gateways or real-time bidding systems.

2. **Limited Execution Time**

AWS Lambda imposes a maximum execution time of **15 minutes** per function. While this is sufficient for many use cases, it becomes a limitation for long-running tasks like large-scale data processing, video encoding, or complex machine learning models that require extended processing time. These types of workloads might run into the time limit and fail to complete, making Lambda unsuitable for certain scenarios. If an application requires more than 15 minutes of continuous computation, developers will need to consider alternative services, such as **EC2 instances** or **AWS Elastic Container Service (ECS)**, where they have more control over execution time. While serverless is great for short, burst-like tasks, the 15-minute cap means it's not the best choice for everything.

Use Case: A company processing large datasets or running data-intensive workflows (such as scientific computations or machine learning training) may find that Lambda's 15-minute limit is insufficient. For such scenarios, they would need to use EC2 or a similar service that doesn't have time restrictions.

3. **Vendor Lock-In**

A major concern with serverless computing is **vendor lock-in**. When you build applications on AWS Lambda, you often become dependent on AWS's specific set of tools and services (like S3 for storage, DynamoDB for database, and API Gateway for managing APIs). This reliance on AWS's ecosystem can make it difficult to migrate to another cloud provider or move to an on-premise solution in the future.

Use Case: A company that builds a serverless application on AWS Lambda, using tightly integrated AWS services like S3, DynamoDB, and CloudWatch, might struggle to move to another cloud platform like Google Cloud or Azure. The cost and effort of re-architecting their system can deter them from migrating, even if another provider offers better features or pricing.

4. **Monitoring and Debugging**

Traditional server-based applications allow developers to monitor and debug their systems with more granular control. In serverless environments, much of the infrastructure management is abstracted away, making it more challenging to diagnose issues. Developers can't directly access the underlying servers, which limits their ability to troubleshoot problems like performance bottlenecks or unexpected failures. In AWS Lambda, developers rely on tools like **AWS CloudWatch** for logging and monitoring, but these tools can be more complex to set up and use than traditional monitoring systems. CloudWatch provides log insights and performance metrics, but developers might find it harder to pinpoint specific issues due to the level of abstraction in serverless environments. Additionally, debugging in serverless can be more difficult because serverless functions are stateless, making it harder to track and reproduce errors across multiple invocations.

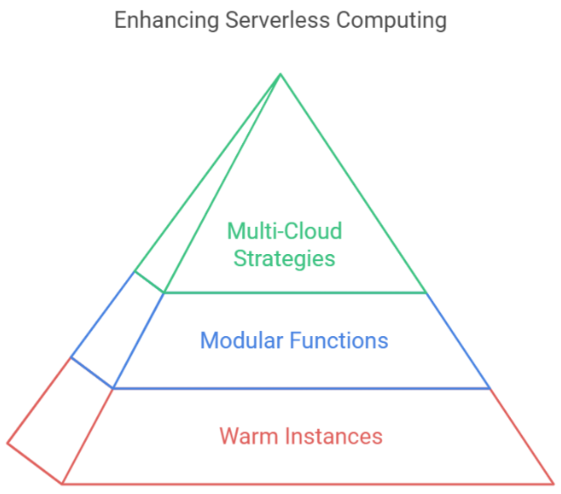
Use Case: A development team running a serverless web application might find it challenging to track down a bug that occurs intermittently. Since Lambda functions don't provide direct access to servers, they would have to rely heavily on CloudWatch logs, which may not give them enough detail about what went wrong in certain cases.

5. Security Concerns

While AWS manages the infrastructure security for serverless functions, there are still **security challenges at the application level**. In serverless computing, it's crucial to configure AWS Identity and Access Management (IAM) properly to ensure that functions have the correct permissions to access other AWS resources. Misconfigured IAM roles could lead to security vulnerabilities, such as unauthorized access to sensitive data or privilege escalation. Additionally, serverless applications are often event-driven, and they can be triggered by a variety of events, such as an HTTP request, a database change, or a file upload. Ensuring that only authorized events can trigger a Lambda function is essential to avoid exploitation. As the serverless model continues to evolve, security best practices are still being developed, which means that developers need to be cautious about securing their serverless applications effectively.

Use Case: An organization deploying a serverless application that interacts with sensitive data (e.g., a financial application) must carefully configure IAM roles to ensure that only authorized services can access the application's resources. If an attacker gains unauthorized access due to misconfigured permissions, it could lead to a significant security breach.

C. *Proposed Solution*



To address the challenges of serverless computing, we propose a comprehensive solution that tackles cold start latency, execution time limits, and vendor lock-in. To minimize cold start delays, techniques like provisioning warm instances or using AWS Lambda's Provisioned Concurrency feature can ensure functions are ready to run immediately, improving performance for time-sensitive applications. For long-running tasks, breaking them into smaller, modular functions that fit within the execution time limits, or using AWS Step Functions for managing complex workflows, can help overcome limitations. To avoid vendor lock-in, businesses can adopt multi-cloud strategies or use abstraction tools like Kubernetes-based frameworks (e.g., K native) to build serverless functions that work across multiple platforms. This approach ensures greater flexibility, allowing businesses to switch providers as needed. Overall, these strategies improve the performance, scalability, and adaptability of serverless computing in modern cloud architecture.

V. Conclusion

In conclusion, serverless computing provides an efficient and cost-effective way to build and run applications, but it also presents challenges that need to be addressed for broader adoption. By applying strategies like reducing cold start times, handling long-running tasks with modular functions, and avoiding vendor lock-in through multi-cloud approaches, businesses can fully leverage the benefits of serverless architectures like AWS Lambda. As cloud technology evolves, addressing these challenges will be key to maximizing the potential of serverless computing in modern cloud-based applications.

References

[1]. Baldini, I., et al. *(2017). "Serverless Computing: Current Trends and Open Problems." IEEE International Conference on Cloud Computing (CLOUD). DOI: 10.1109/CLOUD.2017.30*

[2]. Chandramouli, A., et al. *(2019). "Performance Analysis of Serverless Computing in Cloud Environment." IEEE Transactions on Cloud Computing. DOI: 10.1109/TCC.2019.2904870*

[3]. Lin, J., et al. *(2019). "Serverless Computing: A Taxonomy and Performance Evaluation." IEEE Transactions on Cloud Computing. DOI: 10.1109/TCC.2018.2871360*

[4]. Zaharia, M., et al. *(2017). "Serverless Computing: A New Paradigm for Cloud Applications." IEEE Cloud Computing. DOI: 10.1109/MCC.2017.29*

[5]. Soni, P., et al. *(2020). "An Empirical Study of Serverless Computing: Usage Patterns and Performance." 2020 IEEE International Conference on Cloud Computing (CLOUD). DOI: 10.1109/CLOUD47324.2020.00026*

[6]. Chong, A. Y. L., et al. *(2018). "Serverless Functions in Cloud Computing: A Performance Study." IEEE Cloud Computing. DOI: 10.1109/MCC.2018.2869469*

[7]. Faas, A., et al. *(2019). "Cost Optimization for Serverless Computing: Advances and Challenges." IEEE Transactions on Services Computing. DOI: 10.1109/TSC.2019.2903470*