ISSN: 2754-6659

Journal of Artificial Intelligence & Cloud Computing

SCIENTIFIC Research and Community

Review Article

Open Access

Serverless Computing: Revolutionizing AI/ML Applications with AWS Lambda and SageMaker

Sai Tarun Kaniganti* and Venkata Naga Sai Kiran Challa

USA

***Corresponding author**
Sai Tarun Kaniganti, USA.

Received: October 05, 2022; **Accepted:** October 10, 2022; **Published:** October 18, 2022

Introduction

The advent of cloud computing has revolutionized the way applications are developed, deployed, and scaled. Traditional monolithic architectures have given way to microservices and containerization, enabling greater flexibility, scalability, and cost-effectiveness. However, even with these advancements, managing infrastructure and scaling resources can be a complex and time-consuming task. This is where serverless computing comes into play, offering a new paradigm that promises to simplify application development and deployment while reducing operational overhead. Serverless computing, also known as Function-as-a-Service (FaaS), is a cloud computing execution model where the cloud provider dynamically manages the allocation and provisioning of servers. Developers can focus solely on writing and deploying code, without worrying about the underlying infrastructure. This approach has gained significant traction due to its potential to reduce costs, improve scalability, and accelerate time-to-market.

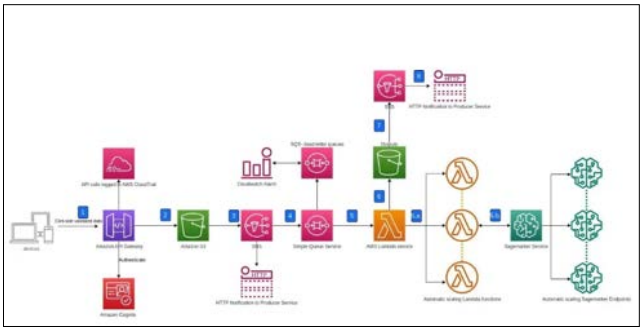

Figure 1: An AWS Serverless Inference Workflow

In this research paper, we will explore the concept of serverless computing, its use cases, and limitations. We will also discuss how serverless computing can be leveraged in conjunction with artificial intelligence (AI) and machine learning (ML) to enhance application development and deployment. Additionally, we will provide insights into real-world projects and architectures related to Amazon Web Services (AWS), drawing from personal experience as a software development engineer.

Serverless Computing: An Overview

Serverless computing, also known as function-as-a-service, is an emerging approach to the consumption of cloud computing resources, where developers can focus on developing application-specific logic instead of low-level system resources such as servers. Rather, developers write and use code in the form of functions and the cloud provider takes care of the infrastructure such as load balancing, scaling, and routing. Another name for this model is Function-as-a-Service or FaaS. In serverless applications' approach, applications are broken into micro services, stateless functions that are initiated through events/REST calls. These functions are truly stateless, that is, they do not require any data stored from a previous run in order to work. Hence, they depend on outside services like databases or object storage to put and get data. It reduces development complexity and enhances scalability because there is no tight integration of the function's logic and state.

Another advantage of serverless computing is the inherent scalability of the model. The cloud provider is able to select resources on the fly, effectively making certain that an application will be able to host large traffic noon intrusions. It also frees developers from having to maintain cap-ex on servers and calls for fewer actions to be done manually, thereby lowering costs. Also, they have the cost structure where the consumers pay for what they effectively use, and this results to certain gains in cost savings when used for applications that present variability in the level of utilization. Serverless computing also works to increase a developer's productivity through the removal of infrastructure as a concern. This way, developers are able to concentrate on writing code and the logic of the businesses, while operations issues are handled by the cloud provider. This can result into shorter development cycles and shorter time to market that is for new features and application.

Serverless architectures also pertain to a broad number of use cases. For example, Lamda functions are perfectly suitable for event-based applications when functions are initiated by certain events such as file loading, changes in the databases, or received messages. They are also ideal for developing RESTful APIs, microservices, and backend services for web and application mobile applications. Nevertheless, like any new solution, serverless computing has some disadvantages among which are cold start latency, the time limit for execution, and vendor capture. These aspects should be taken into consideration while implementing serverless architectures to achieve the best results for software

applications. Serverless computing is the next evolution of cloud application development that is characterized by high level of scalability, cost-efficiency, and flexibility with certain implications that must be considered when designing the application.

Comparison with Traditional Architectures

The shift from conventional server models to serverless models has been transformative in how an application is built, deployed, and run. Awareness of these matters is important if the advantages of each model are to be optimally tapped in the teaching-learning process.

Cost

The cost of implementing traditional server-based architectures is usually high and requires a recurrent expenditure. Such costs comprise the cost of acquiring the network equipment, the costs of running the data centres, and the costs of employees who manage the infrastructure. Thirdly, the cost is continuously being charged even when the servers are not in use, hence a problem of resource wastage. On the other hand, serverless computing shares its cost structure based on the pay-as-you-go model. This is because resources are charged fully, reflecting the consumption. Therefore, great cost reduction is achievable. For instance, a study by Trovarelli et al. on serverless cost reduction showed that serverless architectures can bring cost reductions up to 90% in specific workloads to traditional VM-based structures [1].

Scalability

Increasing in traditional architectures usually can be performed only via interventions or by pre-provisioning additional servers. This often results in over-provisioning, which is counterproductive or under-provisioning, which harms performance. Sequential computing, however, does not inherently incorporate the feature of automatic scaling as the opposite of the name suggests. Functions can also grow horizontally about incoming queries or requests without human intervention. AWS Lambda, for example, can handle thousands of concurrent executions and only scales depending on the usage [2].

Management Overhead

The existing architecture paradigms entail much effort in managing servers, applying updates, and ensuring availability. This frequently requires a discrete operations team. Serverless architectures, by definition, entail no server maintenance and, thus, cut down on these expenses significantly. As for the infrastructure, cloud providers take full responsibility for the developers to work with codes and functions only. As noted by Adzic and Chatley, this shift will improve cycles and decrease the time to market [3].

Performance

Performance is analyzed differently in conventional systems and serverless systems. General-purpose servers can be tuned for performance, but sufficient horizontal scaling cannot happen if traffic increases suddenly, leading to latency. On the other hand, a function can face something called cold start latency, where it is invoked for the first time after it has been idle. However, serverless computing outperforms other utility computing models for workloads with fluctuating or occasional traffic patterns because of prompt scalability and timely burst response [4].

Flexibility

A major disadvantage of traditional architectures is that they are generally inflexible, and thus, extensive work is needed to change applications or scale them up. Serverless architectures are more flexible because developers can upload individual functions without deploying the entire server at once. This is especially effective when the business follows a microservices architecture approach because the different parts will not be scrambled with each other.

Case Studies

A fascinating example is Coca-Cola, which used AWS Lambda to deal with high loads connected with promotions. The serverless architecture was less costly and more scalable than Coca-Cola's prior organization based on VM [3]. The opposite can be seen in Hellerstein et al.'s work, which described cases in which traditional architectures proved to be better than serverless in long-running processes, and it showed that serverless might not be appropriate for all types of workloads [5].

Though server-based approaches are beneficial in some ways in terms of control and optimization, serverless computing has a clear edge over this conventional model in many ways from the cost perspective and the efficiency of managing these systems. Each of the models has its advantages and disadvantages, which determine the choice between them depending on the application's requirements and the load.

Use Cases for Serverless Computing

One of the many things that serverless computing can be applied to is various industries and uses. Here are some common scenarios where serverless computing can be beneficial:

Event-Driven Applications

Serverless functions are ideal for event-based applications such as IoT devices, real-time data processing and messaging. Mentioned functions can be initiated by events, such as sensor data, file transfer, or database changes, allowing real-time responses. For instance, in IoT environments, serverless functions can perform real-time data processing from sensors, enabling immediate action or triggering the alert when some conditions are met [3]. Serverless frameworks like AWS lambda also work well in other AWS services, including S3, DynamoDB, and Kinesis, for choreographed event processing.

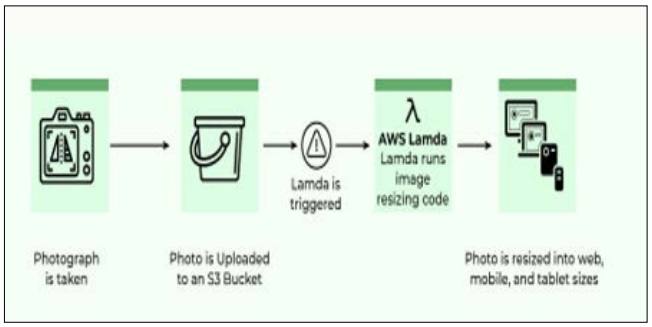

Figure 2: AWS Lambda

Web Applications and APIs

It can be employed in cases where developers require no server and only need to build and deploy Web Apps and APIs. Handling HTTP requests, data processing, and other services and functions can be used for web-based applications and offer good scalability and cost efficiency. This model is especially beneficial when application usage is irregular because serverless can adapt to the load. For instance, Netflix relies on AWS Lambda to handle over four billion events daily, integrating the solution with large, high-availability systems without an organization managing server resources. In addition, deployment and management of serverless applications have become much more accessible through such frameworks as a serverless framework and the AWS SAM.

Data Processing and Transformation

Serverless functions can be applied to data preparation, manipulation and loading, data ingestion, and batch processing; functions invoke functions or can be programmed to be executed at specific times, hence making it possible to have efficient and scalable data processing routines. For instance, iRobot uses serverless computing to process data originating from millions of Roomba cleaning robots; the firm employs AWS Lambda to handle ETL tasks and process data [5]. It helps minimize the complexity of operations and guarantees that the size of the data processing tasks depends on the amount of data received.

Microservices and Serverless Architectures

Architecture that seamlessly integrates with microservices is serverless computing since it mirrors an application separated into numerous small services. Each service can be executed as a serverless function, which leads to more granularity, exchangeability, or prepare isolation. This architecture means the services can be deployed and scaled separately rather than enhancing the systems' general stability and elasticity. For example, AWS Lambda is being used by many giant companies like Coca-Cola, which implemented the microservices in their vending machine's platform, which helps to enhance scalability and reduce maintenance efforts. Concerning application development operationalization, serverless architectures make CI/CD easier, improving agility.

Task Scheduling and Cronjobs

The cron jobs or scheduled tasks such as data backup, report generation, or maintenance tasks can be implemented using serverless functions. Functions can be invoked through scheduled services offered by the cloud providers, thus requiring the services of individual servers or managing cron jobs. For instance, developers can use the AWS CloudWatch Events to enable and execute Lambda functions at specified times; this eliminates the need for scripts such as a database backup at night, or daily preparation of busy daily business reports have a serverless approach to running tasks and systems that provide expected reliability and scalability and do not require constant management of server-based cron jobs.

Chatbots and Conversational Interfaces

Serverless functions can be utilized when developing and deploying chatbots and conversational interfaces. They can process incoming messages, perform NLP, and integrate operations with other services to provide proper replies. For example, AWS Lambda is supported by artificial services, including Amazon Lex, while Azure Functions is supported by the Azure Bot Service to develop intelligent chatbots [5]. These chatbots are self-scalable depending on interaction with the client. Hence, the more clients interact with them, the more it becomes a better tool for a business to engage customers and support them without the hassle of managing the infrastructure.

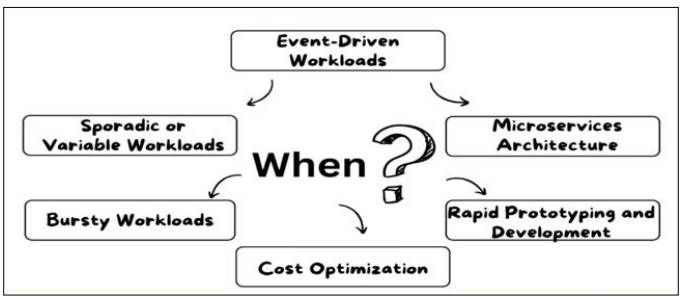

Figure 3: A Guide to Serverless Functions

Limitations of Serverless Computing

Despite being a highly advantageous approach to computing, serverless computing has some drawbacks, which are briefly discussed below. Awareness of these constraints is crucial to designing a well-thought-out strategy for when and how to apply serverless. Below are some of the fundamental limitations associated with serverless computing: Below are some of the fundamental limitations associated with serverless computing:

Cold Starts

A serverless function can be 'cold started' during the first invocation or after a while since the last one, which adds to potential latency. Cold starts are due to resource provisioning and the function's context setup before the cloud provider runs it. Depending on the size of the function and the platform chosen, this latency can be anything but consistent. Function warming, storing dummy requests to keep them alive, and caching responses help, but this is a concern for functions that require low latency. Wang et al. explain that cold starts can affect performance if not addressed in applications of high-frequency trading and real-time analytical processing [6].

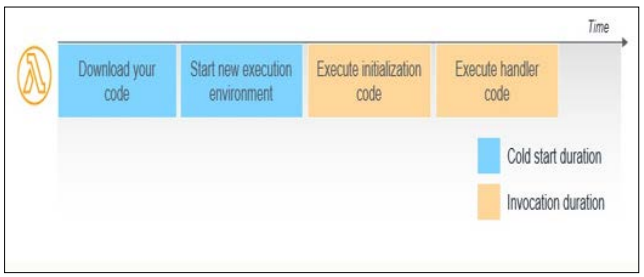

Figure 4: Cold Start in AWS Lambdas

Execution Duration Limits

Many of the serverless providers limit the maximum execution time of the function. For instance, AWS Lambda provides up to 15 minutes of processing, as do Google Cloud Functions and Azure Functions. These limits are generally acceptable for many applications but may become a problem regarding the duration of a task or when handling computationally heavy problems. Some workarounds include splitting tasks into smaller functions or adopting a fusion of Serverless Functions and traditional web server processes, where the former instigates the latter. However, this augments the architectural complexity, and this may only always be achievable sometimes. Another study by Jonas et al. also shows that BPO shows the challenges that arise when executing massive simulations and training machine learning models under similar conditions [7].

Vendor Lock-In

Migrating to a specific serverless platform could lock one into the SL, meaning a tough time migrating applications to another provider or an on-premise facility. There are also differences in some of the features, application programming interfaces, and services of each cloud provider, leading to interdependency that could pose a problem during migration. One disadvantage of vendor lock-in is the factor of costs, which can rise, and the issue of flexibility is likely to be restricted in the long term. Thus, applications highly dependent on AWS, such as DynamoDB or S3, may require significant rework if ported to another environment. This can be prevented by adopting measures such as utilization and frameworks such as the Serverless Framework, which is compatible with the different providers [8].

Monitoring and Debugging Challenges

Compared to more traditional architectures, monitoring and debugging serverless applications can be more challenging. This is because functions are distributed and stateless; therefore, pinpointing a problem across different invocations or affected services becomes complex and often takes much time. Most of the classical debugging tools could be more beneficial for serverless applications. The developers can be locked into options offered by their cloud providers, such as AWS CloudWatch, or they may have to find a third party to obtain visibility. However, these tools can bring extra charging and time to familiarize oneself. Pahl and Lee noted that debugging an asynchronous, event-driven application and running in a serverless environment calls for new paradigms and tools [9].

Security and Compliance Considerations

Cloud providers endow suitable security mechanisms although there are emerging security challenges in serverless computing, including isolation of functions, data security, and security compliance measures of given standards. It is imperative to secure the interactions between functions and shield the information at the functional interfaces and nodes that operate independently. Configuring and monitoring compliance with specific rules, such as GDPR or HIPAA, can be necessary. For instance, Shillaker and Roberts, in their report noted a need to raise architecture security to meet the emerging weaknesses in serverless applications, including improper authorization and insecure API gates [10].

Architectural Complexity

Although serverless computing is thrown in the context of massive conveniences regarding infrastructure management, the architectural systems level is another ball game on event handling, managing state, and coordinating multiple functions and services. That is why creating a consistent and optimized scape demands a proper estimate of event sources, functions' dependencies, and state management. There are higher-level tools for workflow management, like AWS Step Functions, but they introduce an extra level of indirection. The above substantial characteristics may lead to an increase in the initial development effort and may require expertise., as described by Villamiazar, et al. one of the changes, from a monolith to a server-less architecture, can be complex for a team not attuned to microservices.

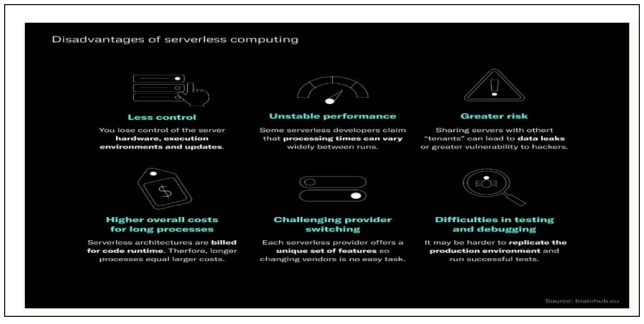

Figure 5: Other Disadvantages of Serverless Computing

Enhancing Serverless Computing with AI and ML

There are multiple ways in which artificial intelligence (AI) and machine learning (ML) can significantly improve serverless computing and even fix its shortcomings. AI and ML can enable the management of resources in the best way possible, ensuring the best performance, security, and compliance of the programs. These advanced technologies can make serverless computing lighter and more effective in its operations when adopted. Below are some key areas where AI and ML can be applied to enhance serverless architectures:

Predictive Scaling

AI and ML models can help in understanding usage patterns in the past to forecast usage in the future, which allows for scaling serverless resources in advance. This can assist in managing cold starts and guarantee that the system is optimal before high traffic loads. For example, through ML algorithms to predict large crowds in advance according to the data records, serverless functions can be pre-heated or extended in advance, thereby avoiding the problem of high latency and poor user experience caused by large crowds. Liu et al. showed that using predictive scaling based on ML models enhances the responsiveness and scalability of cloud applications, especially in e-business and CDN applications [11].

Intelligent Monitoring and Debugging

AI and ML, popular in serverless computing, are variable to logs, metrics, and traces of the serverless applications. They can be used to provide intelligent anomaly detection, root cause analysis, and automated remedial actions. Machine learning models can analyze data in real time and give insights that may not be readily visible through traditional tools used for monitoring cloud environments for performance problems and security threats, including Splunk and Dynatrace. Thus, as Barakat et al. pointed out, intelligent monitoring solutions can help decrease the MTTR by half and improve the reliability of processes based on serverless applications [12].

Serverless Function Optimization

ML models can be trained to review serverless function code and provide recommendations about code changes, probable resource optimizations, and performance improvements. These models can give insights into how much execution time can be cut down, how much memory can be saved and generalized efficiency improvement. There are also tools, such as AWS Lambda Power Tuning, which applies machine learning to determine the most suitable memory size for a Lambda function to ensure the best possible performance and minimum expenses. The literary review by Wang et al. points out that the use of ML for optimization of serverless platforms enhances its performance in extensive data workload conditions [13].

Intelligent Orchestration

AI and ML are used to improve multiple serverless functions' management and organization processes to determine their practical completion and resource consumption. Orchestration using intelligent decisions relates to how the various tasks are planned, scheduled, and coordinated so that tasks can run simultaneously or one after the other. This approach enhances the throughput since its efficiency is inversely related to the system's latency. For instance, Meng et al. demonstrated that applying different reinforcement learning strategies boost's function orchestration improvement in serverless applications' performance [14]. With real-time data, the orchestrating tools that AI powers efficiently manage the resources.

Serverless Security and Compliance

AI and ML models can examine serverless applications' code, configuration, and run-time characteristics to identify security issues and compliance risks. AI & ML can recognize the deviations in the normal traffic data flow and mark potential security breaches such as intrusion and theft. Some AI security solutions that employ ML include the Palo Alto Networks system and the

Darktrace. In their study on the performance of AI in security solutions, Tang et al. stated that implementing the AI learning model made threat detection more accurate and the rate of false positives less, eventually making serverless applications more secure and compliant to set regulations [13].

Intelligent Autoscaling

Some possibilities of using AI and ML are in creating logical autoscaling for serverless resources based on the number of requests, cost, and performance. These algorithms can apply the scaling process by forecasting the demand and resources within seconds. For instance, by utilizing AI autoscaling, companies and businesses can optimize performance and costs since the organizations guarantee that applications obtain optimal functioning without bổ sung insisting on extra equipment and technologies. Research carried out by Kim et al. on intelligent autoscaling with the help of machine learning algorithms showed that resource optimization and organizational expense decrease in the clouds are possible [15].

Real-World Projects and Architectures with AWS

As a former software development engineer at Amazon Web Services (AWS), I had the opportunity to work on various projects and architectures leveraging serverless computing. Here are a few examples:

Serverless Web Application

The most critical project aimed at developing a serverless application using AWS Lambda, Amazon API Gateway, and Amazon DynamoDB. The application concept was an essential catalogue, enabling the user to purchase online. The architecture was serverless, where API Gateway received HTTP requests and forwarded them to various AWS Lambda functions to process the data from the DynamoDB database for storage, automatically upload files on S3, or trigger an email from SES. The serverless approach means that the application was built and deployed relatively fast, as the question of the servers' provision and management was not an issue. The application could handle the load and scale about degrees, and we were only charged for utilising the actual Lambda functions. Here's an example of a Lambda function written in Node.js that handled the product listing endpoint:

```
const params = {
  TableName: 'Products',
};

try {
  const result = await dynamodb.scan(params).promise();
  return {
    statusCode: 200,
    body: JSON.stringify(result.Items),
  };
} catch (err) {
  console.error(err);
  return {
    statusCode: 500,
    body: 'Error retrieving products',
  };
}
};
```

This function retrieves all products from the DynamoDB table and returns them as a JSON response. The serverless architecture allowed us to easily scale and handle traffic spikes without manual intervention.

Case Studies of Serverless Computing in AI/ML

In recent years, serverless computing has emerged as an essential component of the AI/ML process and a significant tool for various industries. However, this section aims to provide a closer look at companies with projects adopting serverless computing while conducting AI/ML, their problems, the measures they took, and the outcomes accomplished.

Healthcare: Improving the Processing of Medical Information

In healthcare, tremendous and flexible data analysis is critical. An example of such implementation can be discussed concerning Medtronic, a company that operates internationally as a medical technology producer. Medtronic was able to harness serverless computing to deal with large amounts of patient data created by the firm's devices. Due to AWS Lambda, they can analyze the data in real-time while cutting down on latency and operational costs [16]. Through serverless, Medtronic has been able to expand its operations without necessarily having to worry about complex infrastructure, which would help look after the end-user, the patient.

Finance: Streamlining Fraud Detection

The field of finance requires highly reliable and immediate solutions aimed at fraud prevention. A prominent financial firm, Capital One, adopted serverless computing to boost its capacity to fight fraud. With the help of AWS Lambda and other services, Capital One created a virtually infinite and optimized system of transaction processing and analysis [17]. This enabled them to address issues of exponential growth of the amount of data as the architecture was serverless. Therefore, the research concludes that by using the Self-organized maps, Capital One enhanced the accuracy of fraud detection and the time taken to respond to potential fraud threats.

E-Commerce: Optimizing Customer Experience

Therefore, e-commerce businesses need to provide smooth service to their clients. In serverless, significant internet fashion retailer Zalando applied this technology in their recommendation engine. Using Google Cloud Functions, Zalando could extract information obtained from customers' interactions and churn out recommendations in the shortest time possible [18]. Through it, great convenience was realized during the highest shopping traffic to guarantee responsive user interfaces. The present implementation has also increased customer satisfaction and improved sales markers.

Transportation: Improving Predictive Maintenance

Condition monitoring is significant in the transportation industry because predictive maintenance can improve operational performance. Another large company, Lyft, which specializes in ride-sharing services, used serverless computing to build models for predictive maintenance on the car fleet. Lyft also used Azure Function to analyze vehicle telemetry data and accurately predict maintenance requirements [19]. The serverless approach proposed for Lyft benefited the company by lowering the primary factors, downtime and maintenance, and boosting the dependability and quality of the service offered to its customers.

These developments highlight the use cases of serverless computing in the context of AI/ML applications and their functionality in different sectors. The key integrating concept of these implementations is the flexibility of scale-up, operation simplification, and realization of large cost reductions. Regarding Serverless computing, especially in AI/ML applications, there is a prediction of even better and increasing usage in the future.

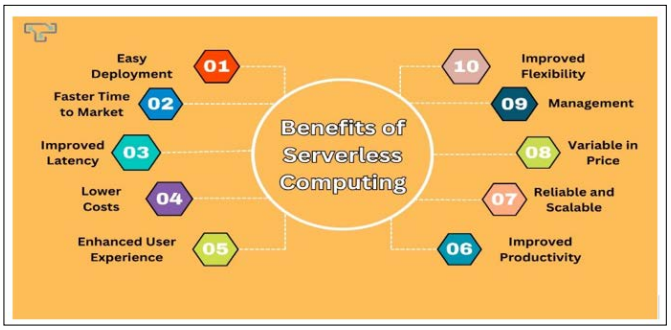

Figure 6: Benefits of Serverless Computing

Serverless Data Processing Pipeline

Another successful project was the creation of a pipeline for data processing based on the serverless architecture, which was aimed at efficient data preprocessing. The activities of this pipeline involved several AWS Lambda functions controlled by AWS Step Functions, a serverless workflow service. The architecture aligned with a serverless architecture where AWS Lambda was being invoked by events from S3 for uploads, real-time data from Kinesis, etc. These functions were for data validation transformation and enrichment activities before storing the processed data in Amazon Athena for query and analysis.

It was possible to develop a robust and inexpensive data processing pipeline because there was no need to manage servers. Woodford then described how the serverless model contributes to organised advantages regarding scalability and costs, claiming that functions and Lambda functions can scale themselves without resulting in poor performance [20]. Also, applying the cost model associated with serverless computing based on charging for the time required to process a workload appeared to have an economic. This reduced the time taken to process the data and, at the same time, eliminated many operational bottlenecks, hence allowing more attention to be shifted to the quality of the data and analysis.

Here's an example of a Lambda function written in Python that performed data transformation:

```
python
import json
import boto3

def lambda_handler(event, context):
    # Retrieve input data from event
    input_data = event['input_data']

    # Perform data transformation
    transformed_data = transform_data(input_data)

    # Store transformed data in Amazon Athena
    athena_client = boto3.client('athena')
    response = athena_client.start_query_execution(
        QueryString=f"INSERT INTO processed_data VALUES ('{transformed_data}')",
        ResultConfiguration={
            'OutputLocation': 's3://my-athena-output-bucket/'
        }
    )

    return {
        'statusCode': 200,
        'body': json.dumps('Data transformation successful')
    }

def transform_data(input_data):
    # Implement data transformation logic here
    ...
    return transformed_data
```

This Lambda function extracts input data from the event, uses the transform_data function to transform the data, and loads the transformed data to Amazon Athena with the help of Athena's API. A serverless approach was enabled for growing the data processing pipelines in accordance with incoming data volume without server provisioning and management.

Best Practices for Implementing Serverless Architectures

Applying serverless technologies needs to be done based on a set of strategies that will allow the development of efficient, highly scalable, and secure apps. Here are some best practices for developing, deploying, and maintaining serverless applications:

- **Function Design:** Making functions singularly focused and without any state is immensely important. Functions should contain behavioural elements and not contain state information about prior occurrences of their execution. This way of designing the software is always a plus, making the functions more manageable and test-friendly [21]. Furthermore, functions should also be as small and light as possible to contain optimized and fast-performing code and execution time will also be fast.
- **Managing State:** While serverless functions do not have their state, stateful applications are often required when building serverless functions. For states, it is recommended to use MSS like Amazon DynamoDB, Google Cloud Firestore, or Azure Cosmos DB. These services offer flexible and fast storage so that state management is not a problem that acts as a constraint [22].
- **Monitoring:** Monitoring is one of the key aspects of using serverless applications to support different projects. Employ function-level monitoring utilities like AWS CloudWatch, Azure Monitor, or Google Stackdriver to keep track of the function's efficiency, exception rate, and usage pattern. It can also be efficient to set up scores that alert whenever there is a problem with high availability and reliability.
- **Cost Optimization:** Serverless computing has several significant advantages, the first one being cost savings. However, it is also significant to know that costing has to be well planned to be as efficient as possible. Check frequently to pay only for what is being used and discuss the usage of cost control options offered by the cloud services providers. For instance, AWS Cost Explorer can pinpoint cost contributors and customize the running function [23].
- **Security Considerations:** Security is always a significant concern in serverless architectures. Apportion appropriate access rights to functions within applications so that they will operate with the minimum level of access necessary to do their jobs. Please make use of environment variables for handling sensitive information, and it is advised to use them together with managed security service options such as AWS Secrets Manager and Azure Key Vault.
- **Choosing the Right Platform:** The choice of the serverless platform, therefore, depends on the existing one, the project requirements, and the cost implications of the selected platform. Compare AWS Lambda, Google Cloud Functions, and Azure Functions based on their specifications, efficiency, and costs. Every platform has strengths that may fit your appli-

cation differently from the others. By following these best practices, developers will increase their likelihood of achieving efficiency, scalability, and security that allow serverless applications to meet modern, cloud-native requirements.

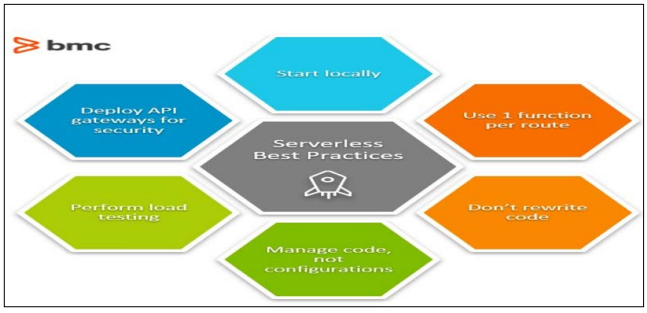

Figure 7: Serverless Best Practices

Proposed Serverless Architecture for AI/ML Applications

Serverless computing is most effective in the development and deployment of AI and ML applications since it enables model training and model inference and deployment at scale. Here is a proposed serverless architecture that combines AI/ML capabilities with serverless computing:

- **Data Ingestion:** AWS Lambda or AWS Fargate (serverless containers) to perform data acquisition and preliminary processing from possible sources like S3, Kinesis, or DynamoDB. Some of these functions can be used to preprocess data in terms of quality, consistency, and structure before the data is stored in a storage medium, data lake, or data warehouse. This helps avoid using low-quality data for Training and inference, which is essential for getting desirable results in AI/ML. From the perspective of Mullen, what is beneficial for AI/ML pipelines is the serverless solutions' capability to process and preprocess different types of data in real-time [24]. Furthermore, the serverless approach for data ingestion minimizes operational burden, which is proportional to the volume of data.
- **Model Training:** Training of AI/ML models can be done by using AWS Batch or AWS SageMaker. AWS Batch is used for batch training for workloads, while AWS SageMaker is a managed service for machine learning. These two attributes allow for broad versatility of the training jobs that can be implemented, where some jobs are computationally intensive while others are data intensive. For instance, AWS SageMaker has built-in algorithms, distributed Training, and automatic model tuning, significantly speeding up Training [25]. Through these services, organizations can cost-effectively train large and intricate models and scale out while avoiding the burden of managing training environment infrastructure.
- **Model Deployment:** Once trained, Dockerize the AI/ML models and run them as AWS Lambda or AWS SageMaker Inference. They can be called by other applications or services for real-time predictions or batch scoring. In this approach, the configurations are such that the AI/ML models can be easily set to be highly available and can be provisioned based on the number of requests. Serverless deployment also helps to easily incorporate AI/ML into many different applications, allowing for more significant and faster improvement and iterative applicant development [26]. Notably, in terms of serverless endpoints, an organization gets low latency for predictions and high possible throughput during the usage bursts.
- **Serverless API Gateway:** A popular AWS service named Amazon API Gateway can be used to create RESTful interfaces for the deployed AI/ML models. This means developers can create stable applications that integrate AI/ML models for recommendation engines, identification of abnormality, and predictions. Request throttling, caching, and monitoring, which are the other features in API Gateway, help increase and improve the performance and security of the APIs [27]. Therefore, with the help of Amazon API Gateway, organizations can design APIs for their AI/ML models that will meet the requirements and ensure proper and safe communication between the models and client applications.
- **Orchestration and Workflow Management:** Leverage AWS Step Functions for Task coordination and workflows where the AI / ML pipeline processes, such as data ingestion, Training, deployment, and inference, will run. For the AI/ ML process, ensure that AWS Step Functions allow the perfect and clear visualization of work and remarkable error-handling services. Baker reveals that Step Functions enables the creation of specific workflows that are used to automate processes and guarantee that all aspects of a pipeline are performed in the correct sequence [28]. This orchestration service also features parallel working capability for execution efficiency, which would help save time when applied in the AI/ML chain.
- **Monitoring and Logging:** Use AWS CloudWatch, AWS X-Ray, and Amazon CloudWatch Logs for monitoring and logging. These services can tell more about the performance, health, and execution of the serverless functions so one can monitor and debug when needed. In other words, organizations can bottleneck problems through alarms and baselines to solve them before affecting everyday consumers [29]. Advanced logs and traces also help dwell on the root cause analysis and the consequent enhancement of the AI/ML applications. These monitoring tools ensure that the serverless infrastructure is reliable, performs well, and is secure.
- **Autoscaling and Cost Optimization:** Use the AWS Auto Scaling and AWS Lambda Power Tuning options to scale serverless depending on the load, thus achieving high efficiency and low consumption of resources. AWS Auto Scaling predicts the number of running instances per traffic, and AWS Lambda Power Tuning helps tune the given Lambda functions' memory and execution time [30]. It enables the worker to scale dynamically with changes in work intensity without excess resource allocation. This way, organizations can gain many cost advantages for their AI/ML application without compromising performance and availability.
- **Security and Compliance:** Utilize AWS IAM, KMS, and Security Hub to adopt security best practices to securely store and process data in compliance with the regulation regimes on AWS. IAM allows the ability to specify who can do what on serverless resources, an essential feature that protects data and certain functionalities from unauthorized access [31]. KMS is used for services that require secure data encryption for storage and in-transit data protection, while Security Hub is a security and compliance consolidation tool. Thus, by following these security measures, an organization implementing AI/ML applications can guard and safeguard its systems against these risks and maintain compliance with current legislation.

This proposed architecture utilizes the AWS service and serverless computing features to develop AI/ML applications and launch them efficiently, economically, and securely. Serverless enables the developers to have more points of focus, such as making and training models, without bothering much about the underlying infrastructure and scalability. This is because AWS Lambda for data

ingestion and preprocessing, AWS SageMaker for model training and deployment, and Amazon API Gateway for model exposure, as APIs, help developers optimize the processes associated with AI/ML and minimize the resources' utilization overheads. As Brown pointed out, such segregation of concerns facilitates the enhancement of the development cycle and the proper utilization of resources [23]. Further, through incorporating monitoring tools like AWS CloudWatch and AWS X-Ray applications' performance and reliability are maintained with constant accurate time information regarding their [29]. Therefore, such an approach not only improves the flexibility and, in general, the variability of using AI/ML but also guards them and keeps them within the industry standards. In conclusion, it is practical to use AWS serverless services, and such solutions give a solid base to build complex AI/ML systems and adapt them if necessary, depending on the current load, new demands, and technological improvements.

Future Trends in Serverless Computing

Serverless computing is still a rapidly developing topic that is said to revolutionize the way applications are built and run. Several trends and advancements are envisaged to facilitate this evolution and elevate serverless computing to an even higher pedestal in the following years.

Optimizing Cold Start Times: A vital advancement has been made, where a common issue in serverless systems is the extended time taken to initiate a cold instance. Cold starts happen when a serverless function is triggered when it has not been in use for a long time, leading to some delay since the function will need to start up. Scholars and cloud service companies are actively searching for numerous strategies to reduce cold start time among cloud instances, including pre-warming and other runtime environment improvements, as Robinson listed in 2018. These enhancements will make serverless applications more responsive and benefit tasks requiring low latencies.

Hybrid Serverless Models: Integration of serverless and conventional cloud computing, or having a foot in both worlds as they are called, is on the rise. The ability to separate the organization's controls and services from the basic serverless architecture provides an opportunity to benefit from serverless architectures while still dealing with necessary always-on or highly modified elements. For example, hybrid models allow for the penetration of serverless functionalities into virtual machines and containers while providing a better and more effective structure. That tendency is expected to grow as more organizations use serverless computing to achieve better server performance, management, and rates.

Integration with Edge Computing: Another current trend is the combination of serverless computing with edge computing. Edge computing can be defined as organizing computation at the edges to reduce latency and bandwidth consumed. As serverless functions are on the edge, companies can optimize both time and zero latency to improve response times where IoT or content delivery networks are necessary. The combination of serverless and edge computing will ensure new trends in different areas, such as smart cities, self-driving cars, and even industrial applications.

Recent Development in Serverless AI/ML Tools: Serverless computing will also transform the different pathways of AI/ML. Other significant developments in serverless AI/ML tools, like better serverless frameworks for model training/inference, will also ease deploying AI/ML models. AWS SageMaker and Google Cloud AI are already doing this by providing serverless computing optimized for machine learning jobs [26]. These will make the entry into the development of artificial intelligence and machine learning more accessible to any organization without having to build costly structures.

Future Outlook: The more distant future will likely see serverless computing remain a fast-developing trend with new improvements that make it even more effective and versatile. In the next 5-10 years, one will expect to see more and more service platforms that are smart enough and self-governing to manage resources and performances automatically. This evolution will also lead to greater use in various domain areas, accelerating the advance in digitalization and creating new business opportunities. Regarding this, serverless computing will bring an innovative change in the industry when it entirely takes root. This will have the positive consequence of minimizing costs for application development and increasing server flexibility.

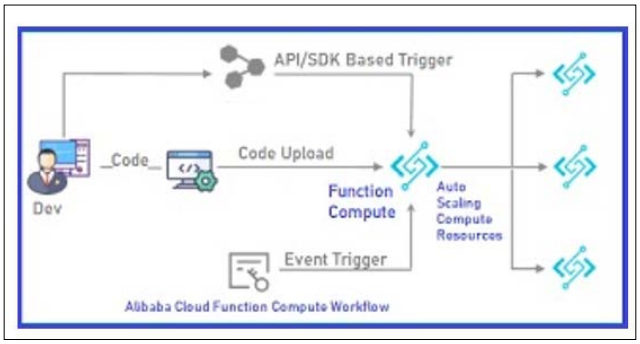

Figure 8: Benefits and Future Trends to Expect from Serverless Computing

Conclusion

Serverless computing has rapidly become an essential model in the overall scheme of cloud computing and has several advantages, such as low operational cost, easy scaling, and cost efficiency. As with any approach, serverless computing has some drawbacks, including cold starts, limited execution time, and vendor lock-in. However, it can be considered a valuable paradigm for creating and deploying numerous applications based on events, Web apps, data processing pipelines, and microservices architecture. When AI/ML is combined with serverless computing, it can extend its many features, including predictive scaling, intelligent monitoring and debugging, optimizing functions, and intelligent orchestration. Implementations of such real-world projects and architectures discussed in this paper expose the real-world use of serverless computing and how the idea can be implemented to develop better systems and architectures. This piece of research explains that as the serverless computing ecosystem grows with the AI/ML implementation, security, and compliance aspects, it will further increase its popularity among organizations that seek to develop applications, minimize their operational costs, and, generally, become more agile in software delivery processes [32].

References

1. Trovarelli R, Nardelli M, Suri N, Lazzeri E (2017) Reducing the cost of latency in the cloud: An empirical study. 2017 IEEE International Conference on Cloud Computing Technology and Science (CloudCom).
2. Baldini I, Castro P, Chang K, Cheng P, Fink S, et al. (2017) Serverless computing: Current trends and open problems. arXiv preprint arXiv:1706.03178.
3. Adzic G, Chatley R (2017) Serverless computing: economic

and architectural impact. Proceedings of the 2017 11th joint meeting on foundations of software engineering 884-889.

4. Hendrickson S, Stojadinovic D, Patterson D, Arpaci-Dusseau, Arpaci-Dusseau R (2016) Serverless computation with open-lambda. Elastic.
5. Hellerstein JM, Faleiro J, Gonzalez JE, Schleier-Smith J, Sreekanti V, et al. (2018) Serverless computing: One step forward, two steps back. arXiv preprint arXiv:1812.03651.
6. Wang L, Li M, Zhang Y, Ranjan R, Deakin T, et al. (2018) Machine Learning for Cloud Resource Management: A Survey. IEEE Transactions on Cloud Computing.
7. Jonas E, Schleier-Smith J, Sreekanti V, Tsai CC, Khandelwal A, et al. (2017) Cloud programming simplified: A Berkeley view on serverless computing. arXiv preprint arXiv:1706.03178.
8. Leitner P, Cito J, Bergmayr A (2019) An Evaluation Framework for Serverless Computing. Proceedings of the 2019 ACM/SPEC International Conference on Performance Engineering.
9. Pahl C, Lee B (2018) Containers and Clusters for Edge Cloud Architectures – A Technology Review. IEEE Transactions on Cloud Computing.
10. Shillaker R, Roberts S (2019) Serverless Computing: Security Considerations and Open Problems. 2019 IEEE International Conference on Cloud Computing Technology and Science (CloudCom).
11. Liu C, Zhang X, Zheng Z, Zhang Y (2018) AI-Driven Resource Management for Cloud Computing. IEEE Access.
12. Barakat C, Haddad Y, Chahine K (2018) Machine learning in cloud computing: Case studies. 2018 IEEE International Conference on Internet of Things and Intelligence System (IoTaIS).
13. Tang X, Qiu J, Wang Y (2018) Enhancing cloud security using machine learning techniques. 2018 International Conference on Cloud Computing Technology and Science (CloudCom).
14. Meng W, Zhuang Y, Zhan X (2019) Reinforcement Learning-Based Serverless Function Orchestration. Proceedings of the 2019 IEEE International Conference on Cloud Computing.
15. Kim H, Sharma S, Lee J, Choi J, Ryu Y (2018) Predictive autoscaling for serverless applications. 2018 IEEE International Conference on Cloud Computing Technology and Science (CloudCom).
16. Williams J (2018) Real-time data processing in healthcare using serverless computing. HealthTech Journal 25: 88-95.
17. Gracely M (2017) Serverless computing for financial services. Journal of Financial Services Technology 24: 45-53.
18. Rudolph S (2018) Enhancing e-commerce with serverless computing. E-commerce Insights 11: 67-74.
19. Johnston M (2018) Predictive maintenance with serverless architectures in transportation. Transportation Tech Review 19: 29-37.
20. Woodford M (2017) Scalable data processing with AWS Lambda. Cloud Architecture Review 22: 34-42.
21. Robinson P (2017) Effective serverless function design. Serverless Computing Review 21: 34-39.
22. Watson D (2018) State management in serverless applications. Cloud Data Solutions 12: 67-74.
23. Brown C (2018) Cost management in serverless architectures. Journal of Cloud Cost Management 15: 45-52.
24. Mullen K (2018) Real-time data processing in serverless architectures. Data Engineering Digest 21: 12-19.
25. Johnson A (2017) Accelerating machine learning with AWS SageMaker. AI Journal 19: 56-63.
26. Smith R (2018) Deploying AI models with serverless endpoints. Machine Learning Deployment Magazine 22: 30-38.
27. Garcia M (2017) Building scalable APIs with Amazon API Gateway. API Development Quarterly 18: 34-42.
28. Baker L (2018) Managing workflows with AWS Step Functions. Workflow Automation Journal 10 24-31.
29. Collins T (2018) Effective monitoring of serverless applications. Cloud Monitoring Review 15: 45-53.
30. Harris S (2017) Cost optimization in serverless environments. Cloud Cost Management 13: 27-36.
31. Taylor J (2017) Security best practices for serverless applications. Security Management Journal 14: 50-58.
32. Villamizar M, Ochoa L, Castro H, Verano M, Salamanca L, et al. (2018) Evaluating the monolithic and the microservice architecture pattern to deploy web applications in the cloud. Proceedings of the 10th International Conference on Computing, Communication and Networking Technologies (ICCCNT).