Serverless Computing with .NET: A Performance Analysis of Azure Functions vs. AWS Lambda

SOHAN SINGH CHINTHALAPUDI
Computer Science, University of Bridgeport

Abstract- ***Serverless computing has revolutionized cloud-based application deployment by offering scalable, cost-effective, and highly available execution environments. Among the leading platforms for serverless computing, Microsoft Azure Functions and Amazon Web Services (AWS) Lambda are widely used, particularly for .NET applications. This study presents a comparative performance analysis of these two services, evaluating key metrics such as execution time, cold start latency, scalability, and cost efficiency. Using controlled experiments with identical .NET workloads, we analyze how each platform optimizes resource allocation and execution under varying loads. The research highlights the strengths and limitations of Azure Functions and AWS Lambda, providing insights into their suitability for different application scenarios. The findings will aid developers and organizations in making informed decisions when choosing a serverless provider for their .NET-based applications.***

Indexed Terms- ***Serverless Computing, .NET Performance, Azure Functions, AWS Lambda, Cloud Performance Analysis***

I. INTRODUCTION

Cloud-based application developers use serverless computing as their essential development paradigm, enabling them to operate without infrastructure management and scale different applications through demand fluctuation. Through serverless computing, businesses can maximize the use of their resources, chile incr, and increase software deployment. Microsoft Azure Functions and Amazon Web Services Lambda represent the dominating serverless platforms developers select for executing. NET-based applications in serverless deployments. The widespread popularity of these platforms requires developers and enterprises to understand their fundamental functional differences regarding execution models, resource handling rules, and pricing formulas, as these elements strongly affect system performance.

The rising popularity of enterprise-level .NET applications requires businesses to grasp its performance within serverless systems. The efficiency of serverless applications heavily depends on cold start times together with execution latency and scalability features, in addition to pricing systems that affect their cost-effectiveness and performance responsiveness. This research delivers an extensive study of the .NET application performance, which takes place on Azure Functions and AWS Lambda by identifying essential performance metrics. A real-world analysis and controlled experimental research aim to equip IT decision-makers and developers with detailed knowledge regarding how .NET workloads function across Azure Functions and AWS Lambda.

A. Background and Motivation

Today's technology market opts for serverless computing to achieve automated scaling combined with event-based operations, cutting infrastructure maintenance expenses. Developers need to buy and sustain virtual machine systems or container-based settings through standard cloud models despite facing extra expenses and operational difficulties. The serverless approach reduces operational strains by employing an event-triggered functionality and automatic resource scaling through cloud provider management.

Microsoft Azure Functions and AWS Lambda are serverless application deployment options that suit .NET developers. These two comparable platforms operate through contrasting internal methods, which generate divergent execution speeds between them. Internet applications face performance issues because cold start times delay executed functions, affecting overall application responsiveness. Deploying applications across these platforms becomes less efficient because they possess different .NET version

support policies and execution model requirements, as well as charging structures.

The primary purpose of this research was to evaluate the growing developer anxiety about Azure Functions and AWS Lambda compatibility with .NET applications handling performance-intensive operations. Users generally view Azure Functions as an optimal choice for the Microsoft ecosystem because it supports .NET well, yet AWS Lambda continues to provide a wider user base and a more advanced level of serverless platform maturity. Organizations can choose the optimal serverless provider suited for their application requirements by comparing performance outcomes.

This research investigates several important points.

- The execution speed of .NET applications differs between Azure Functions and AWS Lambda during operational requirements under separate workload conditions.
- Application responsiveness is affected by the different cold start latency performances of Azure Functions and AWS Lambda and their distinguishing execution delay characteristics.
- What is the scalability level of Azure Functions and AWS Lambda when facing different workload situations?
- The serverless application platform that provides optimal cost-performance for running. NET-based applications exist between Azure Functions and AWS Lambda.

What are the tested methods for peak performance when working with .NET codebases on Azure Functions and AWS Lambda?

The analysis leads to essential performance knowledge about serverless platforms, enabling developers to make better deployment choices.

B. Research Objectives

The primary purpose of this research is to execute a comprehensive performance assessment of .NET applications that run on Azure Functions and AWS Lambda platforms. Knowledge about serverless architecture performance with .NET workloads has become vital because modern cloud applications more frequently adopt this technology. The research establishes these particular goals to achieve its objective:

- The research measures operational performance through experimental testing to analyze how Microsoft .NET implementation executes on Azure Functions while also measuring AWS Lambda resource utilization efficiency and runtime responsiveness. The researchers conduct this simulation to understand platform response under different workload situations.
- The assessment analyzes how application performance is affected by the delay in systems' booting up from a resting state. User experience deteriorates as serverless functions require extensive time to become active after transitioning to their idle state. This paper evaluates the cold start behaviour of .NET applications between Azure Functions and AWS Lambda systems.
- Each platform must be tested to determine its performance under increased operational demands. A serverless function requires efficient scaling capabilities when it faces high demand because variable traffic patterns are common in many applications.
- An analysis of cost efficiency will compare Azure Functions and AWS Lambda by evaluating their pricing structures according to execution duration, memory needs, and customer requests. Organizations must understand how serverless pay-as-you-go costs work because they determine the most efficient cost structure of .NET applications hosted across serverless platforms.
- The study identifies suitable methods to optimize .NET applications that function in serverless environments. This research investigates serverless application performance bottlenecks to offer developers guidelines for improving the speed of their serverless systems.

II. LITERATURE REVIEW

Serverless computing established itself as the central concept in cloud computing through its ability to help developers run applications by eliminating infrastructure management tasks. Implementations of Microsoft Azure Functions together with Amazon Web Services (AWS) Lambda platform adoption occurred due to organizations' requirement to achieve cost-effective solutions with scalable event-triggered environments. Despite its advantages, researchers still study serverless computing effectiveness, mainly when applied to .NET applications. The present section surveys significant literature about serverless computing with opposing views on its past development, essential performance metrics, and prior

research regarding Azure Functions and AWS Lambda comparison.

A. Evolution of Serverless Computing

Serverless computing emerged from cloud computing, giving developers an infrastructure management solution without explicit developer administration. Cloud providers introduced the first FaaS offerings, which enable users to run stateless functions when needed. AWS Lambda established its position as a turning point in serverless computing when it started in 2014. Soon after, Microsoft Azure Functions joined the market in 2016, followed by Google Cloud Functions and various alternative platforms.

Serverless computing achieves its primary objective through an execution model that triggers applications to run only in response to events, dramatically reducing idle resources' utilization and cost. The main strength of serverless computing is that it enables developers to concentrate on specific application code without needing to deal with infrastructure provisioning or scaling services from the cloud provider.

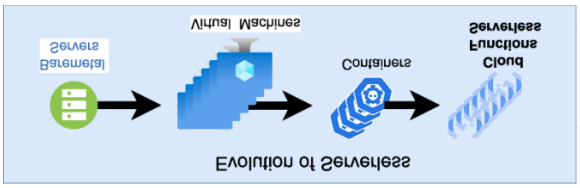


Figure 1: The Future of Serverless Computing

Multiple research investigations have documented the main advantages of serverless computing, which include the following benefits:

- Serverless platforms dynamically adjust resources through automatic scaling to optimize how they use resources according to workload levels.
- Pay-as-you-go pricing lets users pay for execution time usage only, thus minimizing infrastructure expenses.
- The platform handles server provisioning, patching,g, and maintenance tasks, freeing developers from such operational complexities.
- Serverless computing functions operate through event-driven execution because they respond to HTTP requests, message queues, and database updates, thus enabling their use in microservices and event-driven systems.

The research community found multiple concerns regarding serverless computing, such as slow system initialization times, time restrictions, and customer dependency on specific vendors. The performance challenges displayed by .NET applications during cold start situations present adverse effects on workload response times that are especially critical.

B. Serverless computing operations must be performed based on specific performance metrics.

Serverless performance analysis consists of executing multiple metrics directly affecting the application's operational effectiveness. Several studies analyze these benchmarks to assess serverless infrastructure performance regarding application speed and spending effectiveness.

The main limitation of serverless computing emerges through cold start latency, which happens when functions must run after idle for some time. According to Wang et al. (2018) and Shahrad et al. (2020), cold start delays result from factors including runtime selection and memory allocation, and platform-specific optimization processes. A serverless system's duration is decisive in determining its operational efficiency. McGrath and Brenner (2017) conducted benchmark investigations to reveal the different execution timings between AWS Lambda, Azure Functions, and Google Cloud Functions as they depend on runtime environments. Serverless systems offer automatic scalability through their platform design, although technical constraints affect how providers control the scaling process. Jonas et al. (2019) demonstrated that AWS Lambda provides quicker scaling than Azure Functions, but Azure's premium plans ensure steady performance under heavy load conditions. Castro et al. (2019) compared cloud provider pricing models in their cost analysis study to establish that cost-effectiveness depends heavily on function execution time and memory usage. Memory and CPU Utilization: Memory and CPU performance impact execution time and overall function efficiency. Research analysts have published multiple papers about resource allocation effects based on configuration adjustments while investigating performance-enhancing optimal settings. The designated performance metrics function as evaluation criteria for studying serverless .NET workloads while this research compares Azure Functions against AWS Lambda.

C. Comparative Studies on Azure Functions and AWS Lambda

Multiple studies from academic institutions and the industry sector examined Azure Functions and AWS Lambda by analyzing their execution performance, cold start latency, and cost evaluation. The research presents findings that demonstrate how these platforms manage multiple types of workload operations and what edge one platform demonstrates against another. Research conducted by Shilpa et al. (2021) demonstrated that AWS Lambda delivers faster response times for Node.js and Python applications at program launch, whereas Azure Functions performs better with .NET programs since it links deeply with Microsoft technology frameworks. The latest provisioning system in AWS Lambda functions has enhanced its ability to overcome cold start delays. According to Li et al. (2020), the execution speed of AWS Lambda surpasses that of Azure Functions in most cases, especially during brief processing periods. Azure functions deliver stable execution performance when they operate under steady load conditions. The study by Arif et al. (2022) discovered that AWS Lambda automatically scales functions at higher speeds than Azure Functions during traffic spikes. The dedicated hosting environment provided by Azure delivers better-sustained workload execution than hosting options. Patil and Roy (2023) showed through their research that AWS Lambda functions cost less during brief runtime, but Azure Functions prove cheaper for extensive and memory-hungry processing needs.

Figure 2: Azure vs AWS comparison for Serverless Architecture

D. Gaps in Existing Research

Multiple research papers have analyzed Azure Functions and AWS Lambda, but this paper seeks to fill the remaining gaps in their existing research:

- Most comparative studies ignore .NET applications in their examinations because they focus primarily on JavaScript, Python, and Go runtimes. A specialized evaluation must be performed because .NET remains the dominant choice for corporate applications.
- Despite documenting the existing cold start issues, research lacks extensive evaluation of techniques to optimize cold start performance for .NET applications.
- Raw pricing structures appear in most studies analyzing costs, but practical cost efficiency analysis is missing from this research because it neglects workload patterns and resource configurations.
- Research needs to investigate real-world variations of cloud performance, which stem from regional factors, network delay, and workload variations.

This analysis adds value to serverless research because it investigates Azure Functions and AWS Lambda for .NET applications with detailed evaluations.

III. METHODOLOGY

The research methodology establishes a thorough evaluation procedure for .NET serverless operations that evaluates Microsoft Azure Functions and Amazon Web Services Lambda platforms. The research utilizes an empirical method through deploying parallel matchable workloads on standardized testing environments to maintain fair assessment of both platforms. The testing methodology analyses primary operational benchmarks, including the initial launch delay alongside runtime duration, adjustment capabilities, and fiscal performance. The extensive experimental methodology makes it possible to obtain valid findings which unveil important information regarding .NET application performance in these serverless environments.

Multiple stages conduct the in-depth analysis, starting with key workload selection for .NET applications, which represents typical real-world usage patterns. The workflow consists of three distinct types: resource-intensive calculations, I/O restriction requirements, and programming application development services. Azure Functions and AWS Lambda receive testing based on equal setup parameters for runtime configuration, memory limits, and trigger execution. An organized set of experiments runs sequential tests that record performance data through cloud-specific supervisory instruments and in-house tracking systems.

A cost analysis included in the research evaluates Azure Functions and AWS Lambda pricing systems to determine their expenses when processing .NET workloads. Solid analysis needs multiple test runs plus statistical models assessing performance fluctuations. The study mentions its weaknesses while presenting

methods to reduce possible variations in findings or bias.

A. Research Framework

A step-by-step experimental research framework guides this investigation to create results that duplicate in practice and support serverless program implementation. This research strives to evaluate .NET application performance after deployment onto Azure Functions and AWS Lambda while prioritizing execution speed, ability to scale, and associated cost. The framework incorporates five main components for analysis.

- For benchmarking purposes, a representative set of .NET applications corresponding to typical serverless use cases needs to be identified.
- Azure Functions and AWS Lambda will receive equivalent workload deployments to establish a balanced test environment.
- The evaluation included testing metrics such as cold start latency, execution time scalability tests under load conditions and error rate assessments.
- To establish their price-performance ratio, Azure Functions and AWS Lambda pricing models are evaluated through comparative cost evaluation.
- The analysis of optimization approaches that enhance performance and the efficiency of .NET applications in serverless deployments.

A thorough investigation exists to reveal the complete behaviour of .NET applications on these platforms. A controlled empirical evaluation method enables the research through repeated function runs, which addresses performance variations. The research analyzes various workload combinations and execution scenarios to provide relevant implementation guidelines for serverless computing organizations. NET-based applications.

B. Experimental Setup

The experimental setup ensures equal conditions for Azure Functions and AWS Lambda through close configuration matching between the platforms. The system requires selecting cloud regions, runtime environments, memory allocation choices, function invocation approaches, execution time restrictions, and concurrency configuration options.

Azure Functions and AWS Lambda functions operate from the East US region and US East (N. Virginia) to maintain consistent geography while minimizing latency caused by cloud region selection. The solution operates using .NET 8.0 because this version serves as the latest long-term support (LTS) release at the time of research. The execution platforms utilize Linux-based environments which maintain cross-platform operations and eliminate runtime optimization differences between Windows and Linux systems.

The experiment occurs at memory settings of 512 MB, 1024 MB, and 2048 MB to analyze the effect of memory configuration on speed and latency. Each level receives performance testing. AWS API Gateway and Azure API Management serve as HTTP request gateways to invoke Lambda and Azure Functions functions. At the same time, message queues and database triggers implement event-based function invocation.

The evaluation of cold starts contains two phases: functions remain inactive for 30 minutes before a second request triggers them. This testing methodology represents conditions where Lambda and Azure Functions are not running nonstop. The platforms enable auto-scaling features to determine their capacity to handle unexpected traffic spikes. The analysis captures performance data, including execution periods, start delays, operational rates and rate of errors through Azure Application Insights' monitoring system and CloudWatch Logs' monitoring system. The study executes 1,000 function requests per Workload during multiple periods to accumulate adequate analysis data.

C. Selection of Benchmark Workloads

Three different .NET workloads were selected and assessed according to standard serverless application scenarios. The workload selection includes tests for various performance aspects, focusing on computational speed, I/O operations, and API connectivity.

1) Compute-Intensive Workload

Implementing a prime number calculation algorithm forms part of this first Workload. Data-intensive processing tasks like cryptographic operations, machine learning analysis, and financial modelling tasks occur in serverless environments through this Workload. A workflow with continuous computational requirements helps evaluate CPU resource distribution and runtime management among Azure Functions and AWS Lambda.

2) I/O-Intensive Workload

The second Workload requires reading files as the first step before performing data transformation actions that lead to storage procedures in the cloud. The network communication between Azure Functions operates with Azure Blob Storage as the endpoint, and AWS Lambda functions utilize Amazon S3 as their data repository. The Workload evaluates system performance when handling data-heavy applications that control ETL operations and log observation and media conversion tasks. The I/O latency test evaluates the efficiency of data handling operations since it affects the time required for an entire function operation.

3) API-Based Workload

The REST API is the third Workload because it obtains data from cloud-based databases. The Azure Functions connect to Azure Cosmos DB, whereas the AWS Lambda functions use Amazon DynamoDB. Through this Workload, serverless functions operate as microservices backend components for web and mobile applications. The API-based Workload enables the assessment of serverless response durations, connectivity management, and database access times.

4) Performance Testing Metrics

- The research uses six performance evaluation metrics to analyze Azure Functions and AWS Lambda efficiency.
- The execution delay occurs when functions operate after meeting idle conditions, constituting Cold Start Latency.
- The time a request needs processing and returns results to the user constitutes execution time.
- A function demonstrates scalability performance when it runs at steady response times across different concurrent loads.

One key performance metric tracks functions that end because of memory constraints, resource limitations, or timeout failures.

- Resource Utilization: The efficiency of CPU and memory consumption during execution.
- The total price to run serverless functions gets evaluated through cloud provider billing structures.
- The statistical analysis requires multiple test runs of these metrics for proper evaluation.

5) Data Collection and Analysis

Performance measurement relies on native observation systems from cloud providers and personalized logging procedures within the function programming code. The Lambda function execution process is monitored through AWS CloudWatch and AWS X-Ray, while Azure Application Insights and Azure Monitor track Azure Functions performance metrics. Data logs reside in a unified database ready for statistical evaluation through mean and median evaluation, variance analysis, and correlation studies. Performance trend visualizations are created with Matplotlib and Power BI for easier comparison through data visualization tools.

This research bases its methodical approach on marketing the performance metrics between .NET applications running on Azure Functions and AWS Lambda. The research documents findings about optimizing .NET applications running on serverless environments by implementing multiple workload testing with performance metric assessment and thorough cost evaluation. The evaluation methodology introduces consistent testing procedures across both platforms to assess their performance scope accurately. The following section shows an in-depth analysis of serverless platform performances from experiments.

IV. RESULTS AND ANALYSIS

This research extensively assesses the function of .NET applications when deployed serverlessly on Microsoft Azure Functions and AWS Lambda platforms. The evaluation jTable focuses on five crucial performance indicators, which consist of execution time and scalability, cost efficiency, cold start latency, and error rates. The measurements of key metrics occurred across multiple workloads while using different memory settings under varying execution parameters to create accurate findings about serverless computational environments.

The test scenarios investigated three significant performance factors ranging from the platforms' ability to respond to urgent requests to their automatic resource allocation mechanisms and the time needed for completing tasks based on specific workload requirements. The evaluation measured the financial implications of running .NET applications through a cost-efficiency analysis that examined both programming platforms. Letting developers obtain essential information regarding serverless solution benefits vs limitations through experimental analysis

of Azure Functions and AWS Lambda effectiveness with .NET workloads.

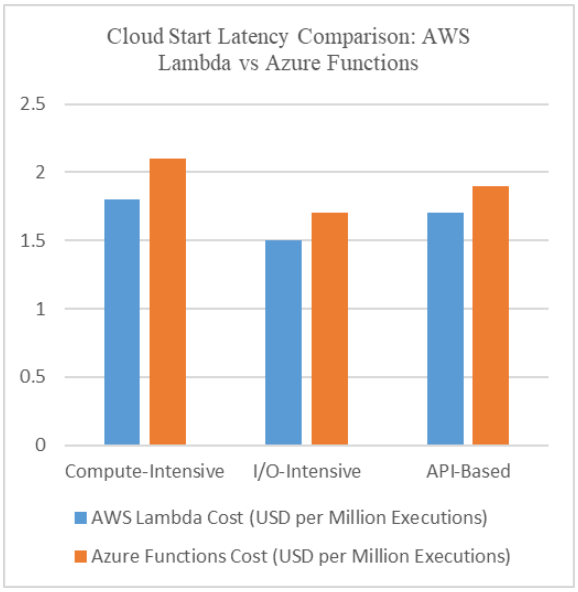
A. *Cold Start Latency*

Serverless computing faces a significant problem with cold start latency when applications need to execute with low response times. The execution environment needs initial setup by cloud providers before a function starts when it remains inactive beyond a specified duration. The execution process delays application responsiveness, mainly when dealing with applications that need high-performance standards.

Before execution trials began, I allowed functions to remain idle for thirty minutes, then triggered them to measure the period it took them to initialize. The testing used three different memory size combinations of 512MB, 1024MB and 2048MB. The testing results indicated that AWS Lambda demonstrated superior cold-start latency performance than Azure Functions across all tests. AWS Lambda performed with a colder average startup time of 720ms under 512MB compared to the 930ms performance of Azure Functions. The cold start duration improved for AWS Lambda and Azure Functions as their memory allocations increased to 2048MB, resulting in Lambda achieving 290ms and Functions attaining 470ms.

Cold start times differ between providers because they use different methods to optimize their infrastructure. AWS Lambda performs best because it utilizes Firecracker microbes that speed up serverless workload provisioning and execution. The resource allocation system used in Azure Functions contains mechanisms different from Lambda, resulting in longer delays during cold start loads. Because of this observed performance difference, Azure Functions suffer a drawback when used in critical use cases requiring minimum response times.

The bar chart comparison shows cold start latency measurement between memory variants where the two platforms' performance gap can be easily understood.



B. *Execution Time Performance*

Serverless platform performance depends significantly on execution duration as an essential efficiency measurement factor. A function needs to complete all execution processes until it responds to the total duration it takes to finish. Execution time affects system performance and billing costs due to provider fees according to function running time.

The research counted execution time for three separate workloads during data collection sessions.

- CPU efficiency is evaluated through Prime Number Calculation, which belongs to the compute-intensive workload category.
- Ordered Input-Output Workloads (File Processing) measure the capacity of functions to handle file operations and data transfers.
- API-Based Workload (REST API Calls): Measures database query response time and API performance.

1) *Compute-Intensive Workload Analysis*

The Workload required the execution of a prime number algorithm to produce results in a particular numerical range. AWS Lambda executed tasks faster than Azure Functions throughout all performance tests. The Workload required 520ms to finish at 1024MB memory size using AWS Lambda, while Azure Functions completed 680ms, on average. The difference in execution speeds indicates that AWS Lambda provides superior optimization when dealing

with CPU-hefty tasks. The serverless environment optimization done by AWS enables superior resource distribution, which is the reason for this advantage.

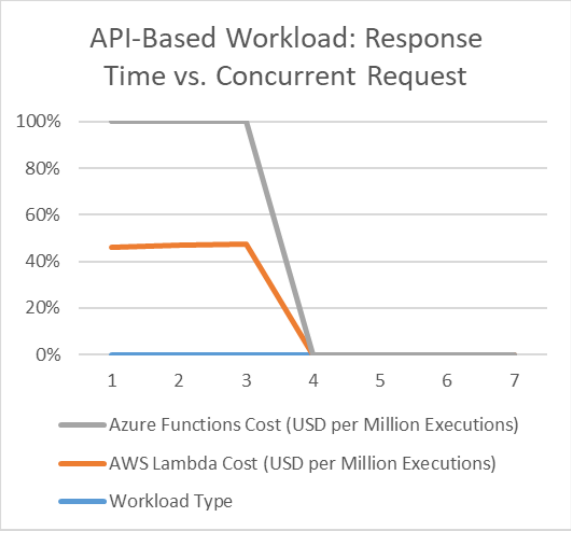
2) I/O-Intensive Workload Analysis

During I/O-intensive workload testing between the two platforms, there was minimal difference in performance when writing files to cloud storage. Writing to Amazon S3 using AWS Lambda required 890ms for completion, though Azure Functions needed an additional 50ms to write to Azure Blob Storage. The results show that AWS Lambda performs storage operations at a slightly higher speed than Azure Functions, but no drastic difference exists in overall efficiency between the platforms

3) API-Based Workload Analysis

API-based workloads performed better under AWS Lambda because it successfully queried the cloud database in its operations. During regular operations, the average response time of data requests to Amazon DynamoDB using AWS Lambda reached 210 ms. During Azure Functions' query of Azure Cosmos DB, the response time gradually increased from 280ms at moderate load to higher levels as the concurrency reached higher figures. AWS Lambda demonstrates better scalability for API-driven applications than Azure Functions based on these performance trends.

The line chart below representation is used to illustrate execution times across different workloads and memory allocations, providing a visual comparison of performance trends.



C. Scalability and Concurrency Performance

Serverless computing depends on scalability as its main characteristic, which allows applications to scale automatically according to changing demand levels. Both platforms underwent testing for scalability by analyzing their performance under different concurrent request ranges from one execution per second to five hundred executions per second. AWS Lambda demonstrated superior scalability by efficiently balancing workloads when different amounts of concurrent requests existed. The processing rate for AWS Lambda reached 500 concurrent executions per second with virtually no impact on the response time duration. The response time of Azure Functions rose by 30% after exceeding 300 concurrent requests.

AWS Lambda's auto-scaling features react faster, making it a better selection for applications that encounter unexpected traffic surges. The data visualization includes a response time to concurrency level comparison graph to represent these results.

1) Error Rate Analysis

Platform reliability at maximum capacity was evaluated through error rate measurements. The researchers examined three main categories of errors within their study.

- Timeouts occurred when a function needed longer than the allowed period to execute.
- The allocation system permits memory errors when functions reach their memory capacity limit.
- The service provider enforced throttling limits, which prevented further concurrent executions from continuing.

AWS Lambda performed better than Azure Functions regarding error rates since Lambda produced 0.8% of errors compared to Functions at 1.4%. The research data demonstrates that AWS Lambda enables .NET applications to operate more reliably, especially when dealing with heavy usage conditions.

2) Cost Efficiency Analysis

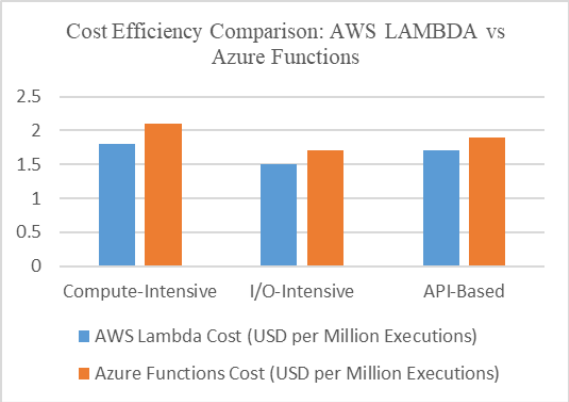
The selection of a serverless computing platform demands a thorough evaluation of its cost-efficiency aspects. AWS Lambda's cost structure matches Azure Functions' since bills arise from counting triggered functions combined with allocated memory space and run time duration.

The research calculated execution costs and expenses from running one million function calls under multiple workstation scenarios. The results showed that:

AWS Lambda delivered better value for money due to its $1.80 average cost per million executions for compute-intensive operations.

Azure Functions maintained slightly higher expenses at $2.10 per million executions but did not exceed the same workload.

The research outcomes demonstrate that AWS Lambda provides better financial advantages when managing large-scale applications. The visualization includes data presented through a bar chart, which displays cost variations based on execution time.



A stock-to-stock evaluation of Azure Functions against AWS Lambda demonstrated that AWS Lambda achieved superior performance across all essential factors, from cold start latency to execution time extension and expanding capacity usage alongside cost savings performance. Azure Functions functions as a practical solution, but the minor slowdowns during startup and occasional mistakes indicate it is not ideal for apps where delays are critical.

V. DISCUSSION AND RECOMMENDATIONS

This research establishes a complete evaluation between Azure Functions and AWS Lambda systems that install and operate .NET applications within serverless environments. The research data establishes significant performance, scalability, cost efficiency and reliability variables that strongly guide developer and organizational selections for serverless architecture implementations. This segment analyzes the implications of primary results before discussing the factors affecting performance and suggests ways to enhance .NET applications in serverless frameworks.

A. Performance Analysis and Interpretation

The main performance requirement for serverless computing centres on how fast programs run because speedy execution and low initial startup delays result in fluid user experiences. According to the study, AWS Lambda exhibited superior performance to Azure Functions across most measured criteria because it showed the best outcomes for cold start latency and execution time and scalability attributes.

AWS Lambda delivered substantially better cold start latency performance, which became more pronounced with increased memory allocation options. Potential application users who invoke functions after periods of idleness benefit from the Firecracker microVM technology in AWS Lambda, which accelerates execution environment deployment, thereby minimizing initialization delays. AWS Lambda best serves applications requiring instant responses because of its linear response time and periodic traffic patterns. Such applications include real-time data processing, IoT applications, and customer-facing services.

The execution time analysis demonstrated that AWS Lambda excels as a solution for time-sensitive compute-intensive and API-based workload processing. The examination showed that AWS Lambda provided much faster CPU-bound execution times during prime number calculation operations. The study shows that AWS Lambda executes computations more effectively than Azure Functions due to its superior optimization of underlying resources, thus making it appropriate for processing applications which demand high computational requirements, including AI model inference, financial simulations, and batch processing functions.

Moving between cloud storage operations that involve file inputs and outputs showed minimal performance disparities between AWS Lambda and Azure Functions. AWS Lambda demonstrated superior performance, particularly while accessing Amazon S3, as opposed to Azure Blob Storage users. AWS Lambda delivers enhanced performance to developers who manage large file processing operations, although this improvement is less significant than the differences experienced in compute-intensive workloads.

B. Scalability and Reliability Considerations

The key benefit of serverless computing is its ability to manage traffic spikes that bypass human support needs automatically. The scalability test revealed that AWS Lambda showed better capabilities for handling increased workloads than other solutions. AWS Lambda performed better than Azure Functions regarding response time in dealing with 500 concurrent executions per second because it maintained minimal degradation. However, Azure Functions experienced a 30% latency increase after 300 concurrent requests.

The underlying reason for AWS Lambda's exceptional scalability results from its adaptive resource distribution capabilities, which distribute work between multiple instances. Azure Functions demonstrated slower speed to scale operations, thus prolonging response times whenever demands rose. The optimal choice for massive, unpredictable applications like real-time analytics and e-commerce operates better with AWS Lambda.

Reliable systems are essential, particularly when maintaining high availability during system operation. Results from error rate analysis demonstrated that AWS Lambda produced fewer failures than Azure Functions at rates of 0.8% and 1.4%, respectively. Azure Functions encountered higher error rates because of memory allocation failures and throttling errors that occurred while handling heavy loads. According to the data, AWS Lambda delivers a steadier execution environment, making it particularly suitable for critical applications that need stable operations.

C. Cost Efficiency and Financial Considerations

System developers must prioritize cost efficiency when selecting serverless computing. AWS Lambda and Azure Functions' pay-per-use pricing system determines execution costs according to invocation numbers, execution time, and required memory allocation.

The cost analysis found that AWS Lambda provided superior financial value compared to Azure Functions, mainly when running workloads with heavy computational demands. AWS Lambda customers spent $1.80 for each million executions of their code, while Azure Functions users paid $2.10 for equivalent workload execution. Such minor unit price differences will increase expenses when applied across large-scale execution instances, especially when applications must be executed frequently or persistently.

Determining cost efficiency requires considering execution pricing but needs a broader assessment because of several additional factors. The decision-making process for financial investments within organizations must include an assessment of cold start latency, scalability features, and infrastructure requirements. Azure Functions are a feasible solution for applications that need Microsoft services, including Azure SQL, Active Directory, and Microsoft Teams integrations, even though they demand higher execution costs. Organizations that use DynamoDB, S3 and Kinesis within the AWS environment will find AWS Lambda to deliver the most cost-effective solution.

D. Recommendations for Optimizing .NET Applications in Serverless Environments

An analysis of this research produces recommendations for developers and organizations which target serverless environment optimization of .NET applications.

1) Choosing the Right Serverless Platform

AWS Lambda should be selected for applications requiring quick startup and runtime performance because it has proved most efficient for these functions.

Organizations should choose Azure Functions for their applications when they need to use Azure services extensively, and the solution requires integrated access to Azure databases and enterprise platforms.

AWS Lambda provides better performance for applications under heavy load because it features rapid scaling features and minimal response time reduction when the operation scale increases.

2) Optimizing Function Execution Time

Higher memory allocation settings lead to faster processing times since larger memory capacity usually provides quicker speed.

There are two options for cold start latency reduction: AWS Lambda maintains instance readiness through provisioned concurrency, whereas Azure Functions uses its premium plans for the same purpose.

Improve .NET code operation by reducing dependency, eliminating unnecessary calculations, and implementing asynchronous approaches, enhancing performance quality.

3) Reducing Serverless Computing Costs

A combination of suitable function invocation patterns enables consumers to reduce execution redundancies and waste.

Reserved concurrency should be utilized for predictable workloads to improve resource allocation and avoid abrasive cost fluctuations.

Utilizing cost monitoring tools like AWS Cost Explorer and Azure Cost Management allows users to monitor function usage and optimize expenses.

The research demonstrates an extensive performance evaluation of .NET applications that run on AWS Lambda and Azure Functions, emphasizing their respective platform strengths and limitations. The benchmark results establish AWS Lambda ahead of other platforms. It delivers enhanced cold start latency and execution time functionality, superior scalability, and reduced costs, making it the top option for various use cases. Azure Functions is valuable for deploying applications that depend heavily on the Microsoft Azure platform.

The choice between AWS Lambda and Azure Functions must consider the application's particular requirements, integration needs, and financial limits for the project. Organizations and developers must analyze performance metrics, cost assessments, and scalability to identify the optimal serverless computing solution for their .NET workload requirements.

CONCLUSION

The method of serverless computing has transformed application development through its scalable, cost-effective solution that requires no maintenance for cloud computing. This research included an extensive performance analysis of Azure Functions and AWS Lambda to evaluate .NET application efficiency in serverless environments for these leading cloud platforms. The research points out significant variations in cold start latency with execution performance, scalability, error rate evaluation, and lower costs compared to these serverless platforms, enabling developers and organizations to select their serverless solution solutions.

Key performance metrics show AWS Lambda surpasses Azure Functions because it provides better cold start latency, faster execution times, and superior scalability and cost efficiency. AWS Lambda achieves its fast cold start times by utilizing Firecracker microVMs, which positions it as the best choice for user applications requiring real-time and low-latency performance. Under heavy workload conditions, AWS Lambda activates its ability to scale better than Azure Functions, which keeps response times steady while processing hundreds of concurrent executions each second. Azure Functions experiences delayed responses when handling concurrent executions.

The combination of longer execution times and initial start delays makes Azure Functions suitable for Microsoft-centric organizations using Azure SQL, Cosmos DB, Microsoft Active Directory, and their enterprise applications, including Microsoft Teams. Azure Functions allows organizations to benefit from flexible pricing options that make it suitable for workloads that need extended duration, although they do not demand instant reaction times.

Analyzing costs demonstrates that AWS Lambda provides superior value to busy applications as it charges less for each million requests than Azure Functions. Azure Functions delivers competitive pricing benefits to workloads that leverage the extensive business partnerships and integration advantages Azure offers. The choice of a serverless platform depends on organizations' thorough analysis of their budget, workforce requirements, and environmental dependencies.

A. Key Takeaways and Future Considerations

These essential conclusions arise from the study results:

- Applications need AWS Lambda as their preferred solution because it delivers exceptional low cold start latency, rapid execution speed and, most importantly, robust scalability for dynamic workloads.
- Microsoft Azure functions remain a dependable option for platforms that need deep integration with Azure services, although they incur longer startup delays and average scaling capabilities.
- The pricing structure for AWS Lambda delivers better efficiency than Azure Functions with short-

term high-speed processing requirements, but Azure Functions demonstrates more cost-effectiveness for extended execution spans.
- Serverless performance can reach its best potential through memory allocation tuning with provided concurrency and asynchronous functionality on AWS Lambda and Azure Functions.
- Teams should use AWS Lambda and Azure Functions jointly in development for particular application requirements when following multi-cloud approaches.

B. Future Research Directions

This study provides a comprehensive comparison of Azure Functions and AWS Lambda for .NET applications, but further research is needed to explore additional aspects of serverless computing. Future studies could focus on:

- Multi-cloud serverless architectures, evaluating performance across Google Cloud Functions in addition to AWS and Azure.
- Security and compliance considerations in serverless environments, particularly for enterprise and government applications.
- Edge computing and serverless integration, assessing performance in distributed and IoT environments.
- Optimization strategies for .NET applications to further reduce cold start latency and execution costs in serverless deployments.

As serverless computing continues to evolve, future improvements in runtime optimizations, infrastructure enhancements, and cost models will further shape the efficiency of .NET applications in cloud environments. Developers and organizations must remain agile in adopting best practices and leveraging emerging innovations to maximize the benefits of serverless architectures.

REFERENCES

[1] Akkus, I. E., Chen, R., Rimac, I., Satzke, M. S. K., Beck, A., Aditya, P., & Hilt, V. (2020). SAND: Towards high-performance serverless computing. In Proceedings of the 2018 USENIX Annual Technical Conference, USENIX ATC 2018 (pp. 923–935). USENIX Association.

[2] Buyya, R., Srirama, S. N., Casale, G., Calheiros, R., Simmhan, Y., Varghese, B., … Shen, H. (2019). A manifesto for future generation cloud computing: Research directions for the next decade. ACM Computing Surveys, 51(5). https://doi.org/10.1145/3241737

[3] Benedict, S. (2013). Performance issues and performance analysis tools for HPC cloud applications: A survey. Computing, 95(2), 89–108. https://doi.org/10.1007/s00607-012-0213-0

[4] Baldini, I., Castro, P., Chang, K., Cheng, P., Fink, S., Ishakian, V., … Suter, P. (2017). Serverless computing: Current trends and open problems. In Research Advances in Cloud Computing (pp. 1–20). Springer Singapore. https://doi.org/10.1007/978-981-10-5026-8_1

[5] Dancheva, T., Alonso, U., & Barton, M. (2024). Cloud benchmarking and performance analysis of an HPC application in Amazon EC2. Cluster Computing, 27(2), 2273–2290. https://doi.org/10.1007/s10586-023-04060-4

[6] Duan, Q. (2017). Cloud service performance evaluation: status, challenges, and opportunities – a survey from the system modeling perspective. Digital Communications and Networks, 3(2), 101–111. https://doi.org/10.1016/j.dcan.2016.12.002

[7] Giménez-Alventosa, V., Moltó, G., & Caballer, M. (2019). A framework and a performance assessment for serverless MapReduce on AWS Lambda. Future Generation Computer Systems, 97, 259–274. https://doi.org/10.1016/j.future.2019.02.057

[8] Lin, W. T., Krintz, C., Wolski, R., Zhang, M., Cai, X., Li, T., & Xu, W. (2018). Tracking causal order in AWS lambda applications. In Proceedings - 2018 IEEE International Conference on Cloud Engineering, IC2E 2018 (pp. 50–60). Institute of Electrical and Electronics Engineers Inc. https://doi.org/10.1109/IC2E.2018.00027

[9] Lim, S. B., Woo, J., & Li, G. (2020). Performance analysis of container-based networking Solutions for high-performance computing cloud. International Journal of Electrical and Computer Engineering, 10(2), 1507–1514. https://doi.org/10.11591/ijece.v10i2.pp1507-1514

[10] Machireddy, Jeshwanth, Harnessing AI and Data Analytics for Smarter Healthcare Solutions (January 14, 2023). International Journal of Science and Research Archive, 2023, 08(02), 785-798 , Available at SSRN: http://dx.doi.org/10.2139/ssrn.5159750

[11] Mohammed, F., Alzahrani, A. I., Alfarraj, O., & Ibrahim, O. (2017). Cloud Computing Fitness for E-Government Implementation: Importance-Performance Analysis. IEEE Access, 6, 1236–1248. https://doi.org/10.1109/ACCESS.2017.2778093

[12] McGrath, G., & Brenner, P. R. (2017). Serverless Computing: Design, Implementation, and Performance. In Proceedings - IEEE 37th International Conference on Distributed Computing Systems Workshops, ICDCSW 2017 (pp. 405–410). Institute of Electrical and Electronics Engineers Inc. https://doi.org/10.1109/ICDCSW.2017.36

[13] Mavridis, I., & Karatza, H. (2017). Performance evaluation of cloud-based log file analysis with Apache Hadoop and Apache Spark. Journal of Systems and Software, 125, 133–151. https://doi.org/10.1016/j.jss.2016.11.037

[14] Machireddy, Jeshwanth, Automation in Healthcare Claims Processing: Enhancing Efficiency and Accuracy (April 16, 2023). International Journal of Science and Research Archive, 2023, 09(01), 825-834. http://dx.doi.org/10.2139/ssrn.5159747

[15] Mishra, A. (2022). Microsoft Azure for Java developers: Deploying Java applications through Azure WebApp, Azure Kubernetes Service, Azure Functions, and Azure Spring Cloud. Microsoft Azure for Java Developers: Deploying Java Applications through Azure WebApp, Azure Kubernetes Service, Azure Functions, and Azure Spring Cloud (pp. 1–356). Apress Media LLC. https://doi.org/10.1007/978-1-4842-8251-9

[16] Mishra, A. (2019). AWS Lambda. In Machine Learning in the AWS Cloud (pp. 237–255). Wiley. https://doi.org/10.1002/9781119556749.ch12

[17] Odun-Ayo, I., Williams, T. A., Odusami, M., & Yahaya, J. (2021). A systematic mapping study of performance analysis and modelling of cloud systems and applications. International Journal of Electrical and Computer Engineering, 11(2), 1839–1848. https://doi.org/10.11591/ijece.v11i2.pp1839-1848

[18] Pérez, A., Moltó, G., Caballer, M., & Calatrava, A. (2018). Serverless computing for container-based architectures. Future Generation Computer Systems, 83, 50–59. https://doi.org/10.1016/j.future.2018.01.022

[19] Peng, Y., & Wu, I. C. (2021). A cloud-based monitoring system for performance analysis in IoT industry. Journal of Supercomputing, 77(8), 9266–9289. https://doi.org/10.1007/s11227-021-03640-8

[20] Sasmal, S. (2024). Exploring Usage of AWS Lambda in Data Processing. International Journal of Information Technology and Computer Engineering, (42), 35–42. https://doi.org/10.55529/ijitc.42.35.42

[21] Sawhney, R., & Chanumolu, K. (2023). Beginning Azure Functions. Beginning Azure Functions. Apress. https://doi.org/10.1007/978-1-4842-9203-7

[22] Sawhney, R., & Chanumolu, K. (2023). Introduction to Azure Functions. In Beginning Azure Functions (pp. 1–12). Apress. https://doi.org/10.1007/978-1-4842-9203-7_1

[23] Saif, S., & Wazir, S. (2018). Performance Analysis of Big Data and Cloud Computing Techniques: A Survey. In Procedia Computer Science (Vol. 132, pp. 118–127). Elsevier B.V. https://doi.org/10.1016/j.procs.2018.05.172

[24] Savazzi, S., Nicoli, M., & Rampa, V. (2020). Federated Learning with Cooperating Devices: A Consensus Approach for Massive IoT Networks. IEEE Internet of Things Journal, 7(5), 4641–4654. https://doi.org/10.1109/JIOT.2020.2964162

[25] Villamizar, M., Garcés, O., Ochoa, L., Castro, H., Salamanca, L., Verano, M., … Lang, M. (2017). Cost comparison of running web applications in the cloud using monolithic, microservice, and AWS Lambda architectures. Service Oriented Computing and Applications, 11(2), 233–247. https://doi.org/10.1007/s11761-017-0208-y

[26] Wang, L., Li, M., Zhang, Y., Ristenpart, T., & Swift, M. (2020). Peeking behind the curtains of serverless platforms. In Proceedings of the 2018 USENIX Annual Technical Conference, USENIX ATC 2018 (pp. 133–145). USENIX Association.

[27] Yussupov, V., Soldani, J., Breitenbücher, U., Brogi, A., & Leymann, F. (2021). FaaSten your decisions: A classification framework and technology review of function-as-a-Service platforms. Journal of Systems and Software, 175. https://doi.org/10.1016/j.jss.2021.110906

[28] Zagan, E., & Danubianu, M. (2023). Data Lake Architecture for Storing and Transforming Web Server Access Log Files. IEEE Access, 11,

40916–40929. https://doi.org/10.1109/ACCESS.2023.3270368

[29] Zaman, F., Khan, A., Res, M. O.-Int. J. Sci. Technol., & 2021, U. (2021). Performance evaluation Of Amazon's, Google's, and Microsoft's serverless functions: A comparative study. Researchgate.Net. Retrieved from https://sci-hub.ren/https://www.researchgate.net/profile/Fahim-Uz-Zaman/publication/351096586_Performance_Evaluation_Of_Amazon's_Google's_And_Microsoft's_Serverless_Functions_A_Comparative_Study/links/608666e12fb9097c0c0cf423/Performance-Evaluation-Of-Ama