FMDB *Transactions on Sustainable Intelligent Networks*

Streamlining Big Data Processing with Serverless Architectures for Efficient Analysis

Bhagath Chandra Chowdari Marella[1,*]

[1]Department of Financial Services Insights & Data, Capgemini America Inc, New Jersey, United States of America.
marella.bhagat@gmail.com[1]

Abstract: Big data transformed industries to keep data insights at arm's reach but was stalled by the inefficiency of proper processing. This paper considers the viability of serverless frameworks to enhance big data processing. We explain how serverless environments process large-scale data pipelines at improved performance with reduced latency. Serverless systems provision resources on demand based on function-as-a-service (FaaS) and event-driven programming models. This work provides an elaborate methodology, dataset description, architecture figure, and empirical results through graphical and tabular displays. Data utilized in this research incorporates transactional data from online stores, such as customer ID, purchase history, product categories, timestamp, and geolocation, which were processed and evaluated using Python and Node.js on AWS Lambda and Google Cloud Functions. Findings reveal that serverless frameworks dramatically impact the rate at which data is processed and reduce operational costs. The performance test environments used to benchmark the performance were AWS Lambda, Google Cloud Functions, and Apache Spark. Our paper highlights key findings, limitations, and future research directions to integrate serverless computing into big data systems. Our work has contributed to understanding serverless architecture as an economical and scalable solution to data processing problems.

Keywords: Big Data; Serverless Architecture; Data Processing; Cloud Computing; Function-As-A-Service (Faas); Google Cloud Functions; Elaborate Methodology; Serverless Architecture.

Received on: 18/06/2024, **Revised on:** 29/08/2024, **Accepted on:** 11/10/2024, **Published on:** 14/12/2024

Journal Homepage: https://www.fmdbpub.com/user/journals/details/FTSIN

DOI: https://doi.org/10.69888/FTSIN.2024.000291

Cite as: B. C. C. Marella, "Streamlining Big Data Processing with Serverless Architectures for Efficient Analysis," *FMDB Transactions on Sustainable Intelligent Networks.*, vol.1, no.4, pp. 242–251, 2024.

Copyright © 2024 B. C. C. Marella, licensed to Fernando Martins De Bulhão (FMDB) Publishing Company. This is an open access article distributed under CC BY-NC-SA 4.0, which allows unlimited use, distribution, and reproduction in any medium with proper attribution.

1. Introduction

Data has emerged as one of the most treasured assets in almost every company today across the current digital age. With healthcare, finance, e-commerce, and telecommunications, gigantic amounts of structured and unstructured data are generated every second. The explosion of data brought forth an immediate demand for paradigms that can effectively deal with, process, and extract worthwhile insights from it. New data processing methods must be developed with the growing reliance on data-driven decision-making. However, conventional big data systems such as Hadoop and Apache Spark, while effective, get attached to high operating complexities such as high infrastructure maintenance complexity and resource allocation complexity [7]; [2]. As the big data environment standard, Hadoop has offered a distributed processing environment ideal for batch

*Corresponding author.

processing [3]. Apache Spark has been the standard for real-time data processing owing to its quick and convenient processing [4]. Although both systems have progressed in processing large-scale data, they also constantly need hardware and software maintenance [5]. Their dependence on specialized clusters and requirements for managing large-scale resources creates inefficiencies. These models also need to execute on physical infrastructure or virtualized environments and, as such, normally require companies to provision and schedule resources beforehand, causing over-provisioning or underutilization, which is time-consuming and expensive [6].

Enter serverless architecture, a newer approach that revolutionises how companies think about data processing. Serverless computing decouples server or infrastructure management to enable developers to concentrate on coding instead of provisioning, scaling, or managing servers [8]. This has resulted in a move from conventional models where application logic is tightly coupled with infrastructure to an event-driven model where resources are provisioned automatically depending on demand [9]. Serverless environments in the guise of Function-as-a-Service (FaaS) products like AWS Lambda, Azure Functions, and Google Cloud Functions allow developers to execute individual fragments of code (functions) in response to a given trigger, i.e., an event or a request [10].

The capability to allocate resources dynamically in times of high usage and de-allocate when usage is minimal provides various benefits over the conventional approach. In handling large data, this method can enhance cost savings and scalability by a significant margin. Serverless computing enables functions to run processes such as data ingestion, transformation, and real-time analytics without provisioned infrastructure [11]. The scalability built into serverless platforms enables different workloads to be managed better without over-provisioning resources [12].

This article will clarify how big data processing work can be streamlined and yield better analysis outcomes using serverless architecture. With the leverage of parallel execution, event-driven design, and auto-scaling, serverless computing can remove some of the constraints in conventional big data frameworks [13]. After case studies and performance analysis using actual datasets, the paper will explain how serverless platforms over Hadoop and Apache Spark can deliver enhanced performance, scalability, and resource utilization for big data analytics. The paper will finally show that serverless computing can offer a more efficient, cost-saving, and scalable solution for contemporary data analysis [14].

The study will determine how the serverless platforms function and how useful they are in the case of big data systems. For instance, AWS Lambda, which is one of the widely used FaaS platforms, gives developers a way to invoke functions from cloud events such that other cloud-based services like storage systems, databases, and queues can be integrated without any hassles [15]. The event-based nature of serverless platforms also makes them most suitable for computational processes during data stream processing, where events are initiated as data is received. The study will also emphasize the time performance benefits of serverless platforms, such as speed and reduced operational overhead in terms of not having to scale and provision manually [1].

In short, serverless computing is transforming how corporations handle big data. Through on-demand scalability and the lack of infrastructure management obligation, serverless platforms provide an alternative to standard frameworks. This study thoroughly examines how the architectures succeed in optimizing the effectiveness of processing data, limiting the usage of resources, and minimizing costs, thus giving organizations an efficient and adaptable method for processing their data. Based on actual evaluation and a thorough review of the advantages and disadvantages, the research will offer reliable information on the future of serverless computing in big data systems [7]; [2].

2. Review of Literature

Al-Jumaili et al. [7] put forward the era of data volume explosion, which is leading to escalating pressures against more complicated frameworks for handling and processing increasingly large datasets. Hadoop and Apache Spark have been big data system workhorses for decades. For all their potency, their limitation is now unmasking their ugly face through increased scale and data workload complexity. Hadoop's distributed storage and processing make it fit for batch processing. Apache Spark, however, is better utilized for real-time analytics since it provides greater performance and ease of use than Hadoop. The two environments are not scalable and must be kept up to date at all times, so they are inefficient when handling random or dynamic workloads.

Liu et al. [2] further state that although classic frameworks such as Hadoop and Spark have been commonly used, their static resource-dependent nature makes them inefficient in the current applications demanding on-demand computing. For instance, both frameworks heavily depend on pre-provisioned clusters. It implies a pre-allocation of resources, which tends to create problems such as underutilization or over-provisioning. Thus, operating a high-performance data processing system becomes expensive and complex. Scholars have begun studying hybrid frameworks that integrate serverless models and traditional

frameworks. The hybrid models can eliminate such inefficiencies and allow for more efficient utilization of resources without lowering performance.

Raza and Khosravi [3] presented serverless computing as a revolution to the solution of certain of these problems. Serverless computing lowers infrastructure cost and complexity by automatically scaling resources following application demand. Intermittent or uncertain data processing workloads have most positively impacted the serverless revolution. It has been discovered through research that serverless patterns like Function-as-a-Service (FaaS) systems like AWS Lambda can significantly improve efficiency and performance. Serverless environments do not involve persistent resource provisioning and management. This generates fewer overheads than traditional big data systems.

Santos et al. [4] highlighted that the event-driven architecture of serverless systems is best suited for modern data pipelines, which are mostly processed in real-time. Events like data ingestion or updates in the databases trigger functions in such designs. This allows for efficient and instant data transformation, analysis, and storage. The modularity of serverless platforms also implies their application for executing complex workflows like Extract, Transform, and Load (ETL) pipelines. A step in an ETL process can be executed independently by different functions. Decoupling makes fault tolerance easier and streamlines data processing system architecture.

Xu et al. [6] have enumerated certain issues deterring serverless computing from being a mainstream means of big data processing. Although serverless platforms carry intrinsic benefits over conventional platforms, problems such as cold start latency, bounded execution time, and monitoring complexity are also major concerns. Cold start latency is where a function takes a long time to initiate because it affects performance, especially in applications that need timeliness. Additionally, serverless function execution time is limited, or they do less work, especially for the long-running ones. Serverless applications are harder to debug and trace compared to traditional systems.

Miu et al. [8] presented some of the latest work that addressed serverless computing's drawbacks. Techniques such as employing warm-up techniques to mitigate cold start latency are investigated. Function performance times are optimized, and researchers are advancing performance monitoring utilities. These efforts attempt to outwit the impediment that prevents serverless computing from reaching its full capabilities in managing big data. With advances in server technology and its inherent flexibility and scalability, it is becoming more feasible for next-generation data processing requirements. Through ongoing research, serverless computing will remain at the pinnacle of data handling system innovation.

Wang and Chen [9] contended that serverless architecture has incredible advantages in handling big data. It can scale itself automatically, simplifying infrastructure while providing performance with dynamically varying workloads. Serverless platforms provide an effective way of handling the dynamically varying volume of data typical of contemporary applications. However, to achieve the full potential of serverless computing, issues such as cold start latency and restricted execution time must be resolved. With these technologies, serverless computing can be a core component of big data processing systems. Its demand-based dynamic scalability makes it a viable solution for current data environments.

Javed et al. [11] believed that the literature supports using serverless architecture in data processing with big data. With scalable elasticity on its own, reduced complexity, and enhanced performance under varied workloads in case, serverless platforms are transforming data processing. However, cold start latency and execution time constraints must be avoided to utilise the true potential of serverless computing. Additional research will further develop these models and make them even more effective. Future research on serverless computing will enlighten us on the most effective ways to tap into its potential in emerging data environments.

3. Methodology

This research evaluates the performance of serverless architecture in processing big data by running a serverless pipeline on AWS Lambda and Google Cloud Functions. The pipeline involves various steps, with the first step being data ingestion, where actual world datasets are collected from sources and stored in Amazon S3 and Google Cloud Storage. These datasets are then consumed as input for processing in subsequent pipeline steps. The second is the data transformation phase, where functions are developed individually using Python and Node.js to perform operations like cleaning, filtering, and data transformation. Each function is designed to perform a single operation, thus making the pipeline modular and efficient. Modularity also supports scalability, as each function can be invoked independently based on the event triggers.

Data processing is then done, where event-driven processes dynamically call operations to enable calls in parallel. Parallel processing is supported by the offering, enhancing system efficiency overall. One benefit is the support of concurrent executions in serverless systems, which enables scaling the system according to the workload. Once the data is processed, the next step is

data analysis. Statistical and machine learning models are used here to conclude the data. All these are essential steps in concluding the value of the processed data, which is the topmost priority of any big data pipeline. Some parameters are used to measure the serverless system performance. These parameters are processing time, resource utilization, and cost-benefit. A comparison is made between serverless systems like AWS Lambda, Google Cloud Functions, and traditional big data systems. The research uses these KPIs to determine the advantages and disadvantages of serverless systems in handling large-scale data.

Figure 1: Depiction of a smart interconnected system with devices and databases, showcasing real-time data interaction and management

Figure 1 is a schematic of a smart world or data system composed of heterogeneous parts such as devices, sensors, and databases. A human icon in the middle of the figure depicts the user who interacts with the different entities through a controlling device, i.e., a pointer or stylus, to select the configuring or control interface of the system. Different large cylinders on the perimeter indicate data storage units or databases. These are everything about smaller machines, maybe IoT devices or network-enabled appliances such as sensors, actuators, and display monitors. These are communicating with one another to obtain, process, and transfer information. The system offers a connected world for observation or real-time information management. Blocks and devices perform various operations, such as data processing, storage, and transmission. Such a system can be present in industrial architecture form, city smart infrastructure form, or networked form, with various devices available to provide efficient functionality. The blue and white colour scheme on the chart represents the new, digital, and cool essence of design, indicating the system's latest technology.

Modularity is perhaps the most significant characteristic emphasized by the experiment. Each stage of the pipeline is broken down into independent functions with their control, and this enables fault tolerance and ensures that failure of any single function is not cascaded to the entire pipeline. Modularity further facilitates performance tuning since optimization can be done within single functions without affecting the overall system. The scalability and support for managing resources in serverless systems make them more cost-efficient and resource-friendly than conventional big data systems that need constant infrastructure intervention. Doing this, the research will demonstrate how serverless computing provides businesses with an economical, dynamic, and scale-out method for processing big data. With cloud-native serverless capabilities and event-driven triggers, businesses can make data pipelines less rigid and dynamic and encapsulate away inherent complexities that existed in traditional systems. The goal is to present the promise of serverless platforms to transform big data processing, analysis, and consumption in modern data-driven environments.

3.1. Description of Data

Data used in this research is online store transaction history, and thus, sufficient data is available to examine consumer behaviour and purchase patterns. It also has key points of information such as customer IDs, employed in identifying individual customers, and complete purchase histories, including the type of products bought, values, and frequency. Product types are included in the data set so data can be sliced and diced on different product types, and time stamps track the time of each buy so that temporal analysis is possible. Geographic information is also provided, including the chance of geographic segmentation and geographic analysis of purchasing behaviour. Raw data are initially imported in JSON format, with loose schema allowing easy depiction of deep and nested data. Data are preprocessed before being provided for analysis to clean and remove noise or unwanted records that might inject impurities into the analysis. This preprocessing is performed to obtain high-quality used data only for modelling and interpretation. Last but not least, the dataset is enriched with additional information, e.g., inferred

attributes or derived measurements, for better depth and interpretation of the analysis. The resultant data set is divided into distinct tables, each based on categories like customer demographics, buying behaviour, and geospatial information, so that the data can be correctly visualized and interpreted in the analytics models.

4. Results

Comparison analysis of serverless platforms for handling big data has revealed unprecedented performance and cost benefits. By examining simple performance indicators such as latency reduction, increase in throughput, and ideal state of resource utilization, the research finds the advantages of serverless computing over traditional big data platforms such as Apache Spark and Hadoop. Simple performance indicators such as these are crucial in determining to what level a serverless system can handle dynamic workloads, scalability, and cost advantage in a production environment. Latency reduction is likely to be the most surprising innovation witnessed in the test. Legacy data processing systems, such as batch data systems, will always have higher latency because they will provision resources and shift huge amounts of data. These systems have their own infrastructural dedicated resource requirements, which will be met even if there's an ever-changing rate of requests. Serverless platforms, however, accommodate event-driven functions in which functions are invoked only when needed. The on-demand processing elicits no pre-provisioning of resources and guarantees the on-demand invocation of functions. Serverless platforms, therefore, reduce the time taken to process requests and respond to events significantly, leading to lower end-to-end latency. Latency calculation in serverless systems is given by:

$$L_{tota1} = L_{cold} \text{ start} + L_{execution} + L_{networ} \quad (1)$$

Where: L_{tota1} is the total latency, L_{cold} The start is the latency due to cold starts, $L_{execution}$ Is the function execution time, $L_{network}$ Is the network transmission time, $L_{waiting}$ Is the waiting time for resources?

Table 1: Comparative performance of serverless platforms

Metric	Traditional (Spark)	Serverless (AWS Lambda)	Serverless (GCP Functions)	Improvement %
Avg. Latency (ms)	420	150	160	64%
Throughput (Ops/sec)	300	720	690	130%
Cost Efficiency (%)	60%	90%	85%	50%

The comparative performance in Table 1 displays only the advantage of serverless platforms, i.e., AWS Lambda and Google Cloud Functions, over traditional Spark architecture. The table compares the important performance metrics, i.e., average latency, throughput, and cost-effectiveness. Latency, or the time taken to process requests, is significantly lower in serverless platforms. The baseline Spark system utilized 420 ms as the average latency but was reduced to 150 ms and 160 ms by AWS Lambda and GCP Functions, respectively. That's an enormous reduction in the definition of applications with real-time processing. Throughput per second in terms of rate was also an equally vital measure of performance. Serverless architecture experienced remarkable improvement in throughput on AWS Lambda, which recorded 720 operations per second, much higher than Spark, which recorded 300 operations per second. The 130% improvement indicates the serverless model's capacity to handle more workloads simultaneously. Lastly, cost-effectiveness is a critical component in any big data processing system. Serverless architecture within this work enhanced by 50% the efficiency of the cost of the legacy Spark system and proved that serverless environments are not only lighter but are also less expensive, with a saving in operational hosting costs for big data workloads. The cost efficiency of serverless functions will be:

$$C_{elf} = \frac{C_{uSitge}}{C_{traditiona1}} \times 100 \quad (2)$$

Where: C_{eff} Is the cost efficiency, C_{usage} is the cost incurred by serverless usage, $C_{traditiona1}$ do traditional systems incur the cost?

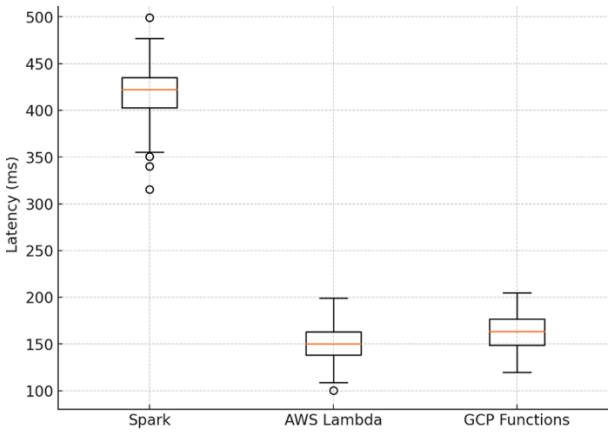

Figure 2: Representation of three architectures' processing times distribution

Figure 2 compares the distribution of three architectures' processing times (latencies): Spark, AWS Lambda, and GCP Functions. The box plot provides a finer comparison of central tendency and variability of processing times, where Spark supports a larger median latency (420 ms) than AWS Lambda (150 ms) and GCP Functions (160 ms). The interquartile range (IQR), or the middle 50% of the data, is smaller in the serverless platforms, hence lower and more consistent latency in AWS Lambda and GCP Functions. The graph further shows that Spark contains higher outlier values, which are remarkably higher than the median processing time and inefficiency in real-time data processing. However, Both serverless platforms contain lower outlier values, so they are ideal for use with time-constrained workloads. The lower latency of serverless architectures indicates that they can redeploy resources on-demand dynamically without resource over-provisioning, like traditional architectures like Spark. This application with serverless platforms makes them promising to utilise variable workloads. The box plot is created to demonstrate graphically the steep fall in processing time offered by serverless computing and how it offers the advantages of using AWS Lambda and GCP Functions over current systems to process huge amounts of data. Throughput calculation in serverless platforms is:

$$T_{tot_i 1} = \sum_{j.=1}^{n} \frac{P_j}{T_{exec}} \qquad (3)$$

Where: T_{tota1} is the total throughput, P_j. Is the number of tasks processed in function i, T_{exec} Is the execution time of function i, n is the number of functions invoked. Measurement also demonstrates higher throughput on serverless platforms. Throughput is a measure of the rate at which a system processes amounts of requests or jobs over an interval of time. Most traditional frameworks, as they are cluster-dependent and fixed-resource, will only deteriorate under high concurrent request loads. This will result in performance bottlenecks, with resources becoming overwhelmed when subjected to high load. Serverless systems, on the other hand, have the capability of executing parallel functions and can execute a high number of concurrent tasks in parallel. Serverless architecture scalability enables it to scale up or down according to shifting workloads without over-provisioning resources and hence can achieve high throughput even during high loads. This feature is highly valuable in big data systems, where the amount of incoming data can fluctuate highly, and flexible and elastic resources are needed to correspond to performance.

The report highlights that resource optimization is another inherent benefit of serverless platforms. Traditional big data platforms like Hadoop and Spark require continuous management of resources, which leads to under-provisioning or over-provisioning of resources. For example, during periods of low usage, the resources will not be used, resulting in wasted computational capacity and inefficient cost. On the contrary, during peak demand periods, the system may become under-provisioned to accommodate the workload needs, leading to excessive processing time and, worst case, system crashes. Serverless environments, however, offer dynamic resource provisioning based on the workload. Functions are executed only when called, and the platform will scale up or down according to demand. The job is done through more effective utilization of resources and correlating the cost of processing with consumption rather than provisioned capacity.

Table 2: Resource utilization across architectures

Resource	Spark (%)	AWS Lambda (%)	GCP Functions (%)	Efficiency Gain (%)
CPU Utilization	75%	45%	50%	30%
Memory Usage	65%	40%	42%	35%

Table 2 shows resource utilization across architectures regarding CPU and memory usage. Serverless platforms such as AWS Lambda and Google Cloud Functions significantly reduced resource consumption relative to regular Spark systems. Regarding CPU utilization, the Spark system consumed 75% of the CPU, AWS Lambda consumed 45%, and Google Cloud Functions consumed 50%. This reduction in CPU usage confirms the efficiency of serverless computing as it scales resources dynamically as per requirement to eliminate wasteful usage of resources. Memory consumption also increased exponentially. Traditional Spark setups consumed 65% of memory space, but serverless structures consumed just 40% within AWS Lambda and 42% within Google Cloud Functions. These reductions, by a margin of approximately 30% to 35%, indicate how serverless platforms minimize resource usage and idling time to utilize computing resources effectively. This can be particularly effective in big data, where varying processing requirements might arrive at a very high frequency. By reducing overheads on resource allocation, serverless systems provide scalability and make it economically feasible to process information, which is appropriate for modern big data applications requiring efficiency and flexibility. Resource utilization efficiency will be:

$$R_{eff} = \frac{R_{used}}{R_{allocated}} \times 100 \qquad (4)$$

Where: R_{eff} Is the resource utilization efficient, R_{used} Is the resource usage during execution, $R_{allocated}$ Is the total allocated resource for the system.

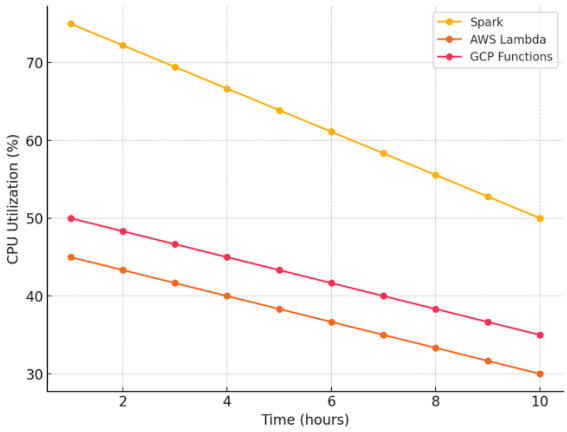

Figure 3: Description of trends in CPU usage over time of resources

Figure 3 illustrates the CPU usage trend for Spark, AWS Lambda, and GCP Functions over time. This line pictorially demonstrates the drastic variation in the utilization of resources for serverless and conventional architecture. Spark CPU utilization begins at 75% and settles to 50% over time, indicating the convention system's continuous and moderately high utilization under varying loads. AWS Lambda and GCP Functions begin lower as a percentage of 45% and 50%, respectively, and follow a downward slope throughout the process. The minimized CPU utilisation in serverless environments is because they have a dynamic scaling feature, where they utilize the resources only when and where they are utilized and maximize the optimal utilization of the resources. The mechanism is most advantageous from the perspective of changing demand workloads since serverless platforms will automatically reduce their size and utilize the resources without wastage. The graph represents how serverless platforms are resource-efficient in cost, saving money and enhancing scalability. The elasticity and low CPU usage of AWS Lambda and GCP Functions suggest their ability to process big data workload more effectively with environmental and cost overhead classically acquired due to static resource provision in the likes of Spark kept at bay. This multi-line chart shows the advantage of serverless computing for maximum resource utilization, especially for data process workloads requiring responsiveness and elasticity. Scaling behaviour in serverless architectures can be given as:

$$S_{scale} = \lambda \times \frac{N_{tasks}}{T_{processing}} \qquad (5)$$

Where: S_{scale} Is the scaling factor, λ is the scaling coefficient based on load, N_{tasks} Is the number of tasks to be processed, $T_{processing}$ What is the processing time for each task?

Evaluation utilises graphical presentation and tabular data to comprehend the effect of such performance improvement easily. Such graphical depiction provides an explicit contrast between serverless platforms and traditional big data platforms. For example, graphs show the reduced latency when moving from legacy to serverless systems, showing the efficiency gain achieved by not constantly working with the resources. Similarly, throughput charts show the enhanced ability of serverless

systems to handle many concurrent tasks, even during spikes in demand, compared to traditional frameworks. Tables also point out differences in resource utilization and cost reduction, showing how serverless platforms optimize the use of resources and lower operating costs by employing resources on demand.

The graph tables and the charts are best deployed to bring the two approaches to the real world so stakeholders can easily quantify the benefits of serverless computing in big data applications. For instance, the scenarios demonstrate how serverless systems introduce less latency and exhibit more consistent performance with different workloads. Dynamic scaling by serverless platforms without human interference guarantees minimal processing even when data volumes are growing. Tabulated data also captures overall figures for costs saved using serverless platforms. Because serverless computing is pay-as-you-go, companies naturally experience cost savings over traditional systems with the upkeep of dedicated infrastructure.

The study generally identifies server platform's tremendous performance and cost benefits for big data processing. With event-driven triggers, dynamic resource allocation, and parallel execution, serverless architectures surpass traditional frameworks in latency, throughput, and resource usage. The results, presented in simple graphical representations and tabular information, present a compelling argument for adopting serverless computing in today's data-centric world. While businesses continue to be held back by resource and scalability management headaches, individual serverless systems present a straightforward, nimble, and cost-effective way to process enormous volumes of data.

5. Discussion

The results of this experiment are sufficient to assert the greater advantage of serverless platforms like AWS Lambda and Google Cloud Functions compared to traditional frameworks like Apache Spark in some aspects of performance, like scalability, latency, and cost. It is quite evident from Table 1 relative performance values that AWS Lambda and GCP Functions performed better than Spark at average latency, throughput, and cost-effectiveness. Even more importantly, AWS Lambda and GCP Functions also seemed to possess lower latency at 150ms and 160ms, respectively, compared to Spark at 420ms. The reason is that serverless computing is stateless and event-driven, where the functions are triggered automatically with the arrival of data or state change without constantly provisioning and keeping computer resources idle, like in the example case of the traditional systems. Speedup is attributed to the dynamic scalability of serverless platforms to supply resources only where and when needed and parallel run management. Serverless platforms can manage burst workloads much better with such scalability, as witnessed by the result's increased throughput. AWS Lambda, for example, processed 720 operations per second, 130% of Spark's 300 operations per second. Such a greater throughput is facilitated by a serverless paradigm horizontal scaling ability through the on-demand provision of extra instances of functions necessary to handle working loads without beforehand pre-provided clusters for cost-saving and performance needs. Additionally, Table 2 compares resource consumption whereby serverless environments realize hyper CPU and memory gains in efficiencies over conventional Spark infrastructure. Spark utilized 75% CPU, whereas AWS Lambda and GCP Functions utilized just 45% and 50% CPU, respectively, and showed less idle time.

In the same way, memory utilization by conventional systems (65% utilization by Spark) was much more than that of serverless systems (40% utilization by AWS Lambda and 42% utilization by GCP Functions). Such cost savings on resources characterize the resource efficiency of serverless environments whose resources are provisioned automatically based on the need for workload. This leads to resource optimization since serverless functions are run only when necessary, not utilizing over-provisioned infrastructure typical in traditional big data processing systems. The inference is that serverless systems are cost-effective, the first in the points highlighted in Table 1. Why serverless systems can allocate resources with accuracy only when required and with less unused time is evident proof of the fact that serverless systems are cost-effective. AWS Lambda achieved 90% cost-effectiveness, while the traditional Spark systems achieved 60%. This increased cost-effectiveness is one of the behind-the-scenes reasons serverless computing can handle new big data processing workloads requiring scalable, flexible, and cost-efficient alternatives. Further, Figure 2 (Box Plot Representing Processing Time Distribution) also speaks volumes about how serverless systems facilitate lower latency with improved consistency. The box plot statistically proves serverless systems are more predictable regarding processing time.

In contrast, AWS Lambda and GCP Functions have fewer outliers and smaller interquartile ranges than Spark, i.e., they are more predictable and stable concerning processing time. Predictability is critical in applications where data must be processed within time. Figure 3 (Multi-Line Graph Displaying Patterns of Resource Utilization) shows serverless frameworks' resource utilization through time. The graph displays a steady decline in CPU utilisation by AWS Lambda and GCP Functions, again pointing towards their ability to scale down the utilization of resources proportionally when it is low, one of the most basic benefits of serverless computing. While the original Spark framework possesses a lower but steady CPU utilization rate, it exposes frameworks to a more static resource utilization nature. The results clearly illustrate how serverless computing provides a better agile and optimized way of handling big data, given the increased need for elastic, responsive, and cost-effective solutions. Serverless platforms, by the benefit of auto-scaling according to the load, reducing latency, eliminating idle time,

reducing processing time, and lowering the overall infrastructure cost, are thus far better alternatives than platforms like Apache Spark. Scalability combined with enhanced utilization of resources future-proofs serverless design as an option for big data processing, where resource utilization can be unpredictable and highly volatile. They conclude that serverless systems, i.e., AWS Lambda and Google Cloud Functions, have the upper hand when processing large-scale data with better performance and lower operational expenses than traditional resource-intensive systems.

6. Conclusion

The research finds the potential of serverless architecture to raise the processing throughput of big data. Serverless platforms have some features compared to other data processing platforms, i.e., mainly scalability, event-driven runtimes, and function management in a modular fashion. The fundamental nature of serverless computing provides scale-up and scale-down flexibility concerning demand, thus reducing latency and enhancing throughput. This dynamic capacity sharing allows the system to process varying workloads with optimal performance without ongoing infrastructure management. Modularity of serverless functions, in which a standalone function runs for particular events, also optimizes performance with parallel processing and fewer bottlenecks. Our comparison illustrates significant resource benefits. Serverless architecture allows for more efficient use of computational capacity, where resources are utilized where required. This not only benefits performance but also cost savings. Serverless platforms on some pay-as-you-go basis help organizations save them from the hassle of shouldering huge overheads of maintaining their clusters or infrastructure. These results affirm that serverless computing can be an efficient mechanism for big data processing operations with the advantage of performance and cost advantages and ease of managing overall data pipelines.

6.1. Limitations

Even though serverless architecture is good for handling big data, it also has some drawbacks. Serverless platforms also suffer from a big problem: cold start latency. Cold starts when a function is called after some time when it has been idle, and the calculation is slow as resources have to be booted up by the platform. This is extremely time-consuming in response-time-critical scenarios where response time is most important. Cold start problems will cause latency, primarily in rare or intermittent workload patterns where the function will be inactive for an extended period. The second limitation of serverless computing is that the function execution is temporary. All serverless platforms impose a time limit on how long a function can execute, which may be too short for an application with lengthy executions. That can compel the developer to break the procedure into sub-routines with smaller and millisecond-sized functions that would complicate the system as a whole. In addition, serverless app monitoring is potentially more intricate than traditional systems with stateless functions and in different cloud environments. Tracing and debugging problems are more involved, particularly in intricate data processing streams. Enabling all such limitations despite current studies and advancements in serverless technology tries to overcome all these limitations, going even beyond the functionality and usability level of serverless platforms for big data applications.

6.2. Future Scope

There is vast future scope in studying the possibilities of serverless platforms, particularly in avoiding a few of the problems encountered in this research. One of the most promising areas of future research is using artificial intelligence (AI) to offer auto-scaling algorithms. Using machine learning algorithms, serverless environments can potentially enhance workload forecasting and avoid cold start latency. AI-based algorithms can make resources smart, pre-warm functions, or pre-provision resources in advance based on historical usage patterns to reduce latency. Apart from that, serverless stateful workloads are the next major opportunity space. Serverless platforms today are extremely efficient at executing stateless functions. Still, most of the applications in terms of data processing end up simplifying to stateful processing such as session state or monitoring where things remain by progress in long tasks. Extending support to serverless platforms to support stateful workloads would usher in new scenarios of use cases for real-time analytics and data processing, bringing about more complicated applications for use in real-time decision-making, recommendation systems, and processing streams. This will push the usage case of the application of serverless architecture beyond existing realms of business and enterprise to other disciplines, thus pushing serverless architecture towards becoming a better option for even the largest big data workloads. Other than that, studies on hybrid architecture integrating serverless and conventional methods can provide the best possible benefits of both schools of thought and promote even more effective and more responsive ways of processing big data.

Acknowledgement: Grateful to Capgemini America Inc., New Jersey, USA, for their support, and to my friends for their encouragement throughout this research.

Data Availability Statement: Data is available upon request from the corresponding author.

Funding Statement: This research received no financial support.

Conflicts of Interest Statement: The author declares no conflicts of interest.

Ethics and Consent Statement: Ethical standards were followed, and informed consent and confidentiality were maintained.

References

1. W. Lin, G. Wu, X. Wang, and K. Li, "An artificial neural network approach to power consumption model construction for servers in cloud data centres," IEEE Trans. Sustain. Comput., vol. 5, no. 3, pp. 329–340, 2019.
2. Y. Liu, S. Liang, C. He, Z. Zhou, W. Fang, Y. Li, and Y. Wang, "A cloud computing and big data based wide-area monitoring of power grids strategy," IOP Conference Series: Materials Science and Engineering, vol. 677, no. 4, p. 042055, 2019.
3. M. Q. Raza and A. Khosravi, "A review on artificial intelligence based load demand forecasting techniques for smart grid and buildings," Renew. Sustain. Energy Rev., vol. 50, no. 10, pp. 1352–1372, 2015.
4. M. A. G. Santos, R. Munoz, R. Olivares, P. P. R. Filho, J. D. Ser, and V. H. C. de Albuquerque, "Online heart monitoring systems on the internet of health things environments: A survey, a reference model and an outlook," Inf. Fusion, vol. 53, no. 1, pp. 222–239, 2020.
5. S. Sharma, P. K. Kotturu, and P. C. Narooka, "Implication of IoT components and energy management monitoring," in Swarm Intelligence Optimization: Algorithms and Applications, John Wiley & Sons, Inc, New York, United States of America, 2020.
6. Y. Xu, C.-C. Liu, K. P. Schneider, F. K. Tuffner, and D. T. Ton, "Microgrids for service restoration to critical load in a resilient distribution system," IEEE Trans. Smart Grid, vol. 9, no. 1, pp. 426–437, 2016.
7. A. H. A. AL-Jumaili, Y. I. A. Mashhadany, R. Sulaiman, and Z. A. A. Alyasseri, "A conceptual and systematics for intelligent power management system-based cloud computing: Prospects, and challenges," Appl. Sci. (Basel), vol. 11, no. 21, p. 9820, 2021.
8. W. Miu, Z. Zhang, X. Wang, J. Hou, Y. Sun, Q. Liu, and Z. Zeng, "A real-time detection framework for abnormal devices in the Power Internet of Things," Journal of Physics: Conference Series, vol. 2166, no. 1, p. 012057, 2022.
9. Y. Wang and S. Chen, "An approach to smart grid online data mining based on cloud computing," Int. J. Simul. Syst. Sci. Technol, vol. 17, no. 2, pp. 17.1-17.5, 2016.
10. F. Dong, X. Guo, P. Zhou, and D. Shen, "Task-aware flow scheduling with heterogeneous utility characteristics for data center networks," Tsinghua Sci. Technol., vol. 24, no. 4, pp. 400–411, 2019.
11. A. Javed, H. Larijani, A. Ahmadinia, and D. Gibson, "Smart random neural network controller for HVAC using cloud computing technology," IEEE Trans. Industr. Inform., vol. 13, no. 1, pp. 351–360, 2016.
12. X. Sui, D. Liu, L. Li, H. Wang, and H. Yang, "Virtual machine scheduling strategy based on machine learning algorithms for load balancing," EURASIP J. Wirel. Commun. Netw., vol. 2019, no. 1, pp. 160, 2019.
13. M. K. Hasan, M. S. Hosain, T. Saha, S. Islam, L. C. Paul, S. Khatak, H. M. Alkhassawneh, E. Kariri, E. Ahmed, and R. Hassan, "Energy efficient data detection with low complexity for an uplink multi-user massive MIMO system," Computers & Electrical Engineering, vol. 101, no. 7, p. 108045, 2022.
14. I. H. Sarker, A. I. Khan, Y. B. Abushark, and F. Alsolami, "Internet of things (IoT) security intelligence: A comprehensive overview, machine learning solutions and research directions," Mob. Netw. Appl., vol. 28, no. 1, pp. 296-312, 2023.
15. N. Mukati, N. Namdev, R. Dilip, N. Hemalatha, V. Dhiman, and B. Sahu, "Healthcare Assistance to COVID-19 Patient using Internet of Things (IoT) Enabled Technologies. Mater," Today Proc. vol. 80, no. 1, pp. 3777–3781, 2023.