International Journal of Applied Mathematics
Volume 38 No. 7s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

# SERVERLESS DATA ENGINEERING WITH AWS LAMBDA AND STEP FUNCTIONS FOR REAL-TIME ANALYTICS

Raghu Gopa[1]
Wilmington University, USA

## Abstract

The ongoing trend, coupled with the challenges of managing large real-time data streams, has transformed modern data engineering by creating the need for systems that are scalable, cost-effective, and highly responsive. Furthermore, the inability to effectively handle dynamic workloads undermines the advantages of infrastructure-driven approaches, often resulting in inefficient resource utilization and delays. Another key element of the paradigm shift is that AWS Lambda and Step Functions are event-driven, elastic, and highly modular, enabling the construction of data pipelines that support real-time analysis. In this paper, the principles of serverless architecture will be presented, where data engineering is guided by these ideologies. AWS Lambda is highlighted as the central processing core, while Step Functions serve as the orchestration core for managing complex workflows. It discusses intrinsic issues, including execution limits, cold starts, monitoring complexity, provider lock-in, and compliance, but also reports deployed applications in finance, IoT, health care, cybersecurity, and personalisation. The future of serverless analytics will be characterized by machine learning, hybrid architectures, edge computing, and enhanced governance frameworks in the future. The objective planning background of the research, which is furthermore factual, is the study that claims that when planned and optimized, both AWS Lambda and Step Functions provide the agility and scalability that organizations demand to make real-time decisions in the modern enterprise.

## 1. Introduction

The past several years have altered the face of data engineering because of the principal convergence of cloud computing, serverless, and real-time analytics. With the increased generation of structured and unstructured information by organizations in volume, older infrastructure-intensive solutions failed to deliver scalability features, agility, as well as cost-effectiveness features desired by organizations. One of such paradigms is that of serverless computing, which is facilitated by services such as AWS Lambda and Step Functions and is changing how analytics pipelines are configured and maintained. Abstracting out the functionality of the infrastructure allows engineers to focus on data processing and orchestration, and displace the functionality of scalable behavior, and the cloud provider standard scalability and availability view [1][2]. The critical point of such transformation is the growing pressure on real-time analytics. Despite its reliability, batch-oriented discomforts create delays that are largely out of place when waiting, such as in cases of fraud identification, faulty maintenance, or personalisation. Instead, real-time pipelines jump to obtain and process data continuously, providing a hint of the ongoing situation. Pixels Serverless architectures and the event-based setting can leverage



Received: August 09, 2025

Volume 38 No. 7s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

on such expectations by evading unused resources when depending on scaling up to match the dynamic traffic [3][4].

Function-as-a-Service (FaaS) AWS Lambda offers functions to be invoked according to a broad series of events, and ingestion and transformation are not a premise in other functionalities as well as they are in Kinesis, DynamoDB, and S3. Since work groups operate as distributed processes, Step Functions facilitate their integration along defined execution pathways. This enables the creation of real-time, adaptive, fault-tolerant, and cost-optimized pipelines for delivering analytics [5][6]. Despite these advantages, serverless adoption is also limited by time constraint issues during execution, cold start, debugging process, and state. The practice of pipelines generally consists of having to collect heterogeneous tasks, satisfy high-level or government restrictions, which are particularly vulnerable to architectural planning [7][8]. In this paper, the role of AWS Lambda and Step Functions will be discussed in the context of serverless real-time analytics pipelines, their architecture foundations, frequent applications, and limitations. It prescribes research and industry practice to show the role of the technologies to erode scalable, resilient, and cost-effective data engineering. With this foundation established, the next section goes through the literature that is available on the serverless paradigm, plus its application in practical use in analytics [9][10].

## 2. Literature Background

A broad appeal underpins the relevance of various frameworks. The growing interest in providing infrastructure-transparent alternatives to overcome the resource intensiveness of traditional systems has led to global recognition of serverless computing and data engineering as effective solutions for delivering responsiveness to high-scale events. It has been repeatedly mentioned in the literature that the streaming analytics pipelines' view outlook has emphasized the ideal suitability of serverless architectures since they are elastic and are conducted in an event-driven fashion. On the other hand, despite their efficiency in application, real-time model Apache Flink and Spark streaming are constrained by cost and operating factors when applied in high-scale topology, owing to the utilization of a permanent cluster [11][12]. Serverless models circumvent such problems by shifting computation to functions that are executed on event occurrences atop infrastructure as opposed to infrastructures that are available long-term.

A difference between a batch and a stream process is one of the themes that recur in the research. Traditional big data systems, primarily designed for batch processing, were effective for analyzing historical information but inadequate for scenarios requiring the urgent processing of real-time data. There are complicated sectors where we are either seeing sub-second responses being more and more vital, like finance, healthcare, cybersecurity, and real-time analytics, which are on the rise. This is supported by the millisecond responsiveness and scalability of AWS Lambda and by Step Functions, which is based on the model to orchestrate stateful, step-by-step workflows [13][14]. It also has a clear integration of the streaming services and distributed storage. Data-streams using Kinesis and S3 can scale up to ingest and back up such data-streams, and so can resilient pipelines frontloaded by Euler plus in tandem with Lambda, without the complexity of a complex infrastructure. Studies on facilitating the ordering of events, retries, and error-catching should, however, be discussed. Step Functions

Volume 38 No. 7s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

can be used to overcome these barriers to provide orchestration capabilities, including error handling, retries, and rollback plans [15][16].

The other excellent concern of research is cost efficiency. Compared to clusters, where there is an always-on cost, serverless systems can, in fact, never incur costs except as per execution, and so are typically used with spiky or intermittent loads. The pay-per-execution billing model will avoid wastage and can test the analytics workflows without huge upfront expenditure [17][18]. Nevertheless, certain limitations persist, as even performance-sensitive pipelines remain vulnerable to cold starts, execution timeouts, and memory constraints. The notion of serverless elasticity hybridization with long-term infrastructure has emerged as a topic of increasing interest in research in order to make responsiveness and reliability compromises [19][20].

Finally, orchestration emerges as a cornerstone of effective pipelines. In its absence, the distributed functions may get out of control and run astray. Step Functions are generally said to be able to support retries and branching and compliance requirements, and provide visibility to workflows. This focus on orchestration might be warranted by the demands to have organized, auditable analytics processes [21][22].

Collectively, the literature indicates that serverless computing is not a space tool that can be used to save costs, but it is a transformation in the design and management of the real-time analytics systems on its side. In this section below is followed by the architectural paradigms by which UI and Step Functions are implemented and scalable, event-driven data processing is thus rendered practicable.

## 3. Architectural Foundations of Serverless Data Engineering

In the context of the explained theoretical background of serverless computing, we would like to take a closer look at architectural concepts that allow making data engineering practical sometimes. All of these concepts are mirrored in AWS, expressed as event-driven design, ephemeral compute-warehouses, and integration of cloud-native services forming and scaling real-time analytic pipelines, all of which shape the creation and scaling of CiDaaS pipelines [23][24].

The fact is that there is no dependence between infrastructure and application logic. Traditional analytics relied on operating clusters, which continued to burn resources where unnecessary. Compared to this, serverless systems just deploy compute on demand, and scale on demand to meet surges of the workload at no cost in idle time [25][26]. Sources a Kinesis as well as S3 and DynamoDB Streams, stateless, and invoke settings and Lambda functions. This stateless performance allows thousands of parallel versions of the functions to process and change events in parallel to ensure almost-instant responsiveness [27]. Orchestration is also significant, and Step Functions orchestrates fine-grained transformations, where in Lambda, they are fine-grained, with the retries and error handling in addition to the logic of branching being internal. This eases the intricacy of operation and guarantees resiliency of the distributed analytics pipelines [28].

Streaming and AWS storage are added to the architecture. One could transfer data to ingestion (Kinesis) and processing (Lambda) and storage or query (S3, Redshift, Elasticsearch) without

Received: August 09, 2025

Volume 38 No. 7s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

bearing the adverse cost of unused processing power, which corresponds to running technologies that are not bottlenecking, on an asynchronous communications basis [29]. Security and administration are also important considerations. Lambda implements the IAM policies, which can be consumed across workflows by Step Functions and can be utilized to impose any other standard that must be adhered to, such as GDPR or HIPAA. This renders the serverless pipelines practically viable even in a setting where controlled data takes preeminence [30]. This is done through the visibility similar to CloudWatch and the history of calling Step Functions to visualize the lagging numbers, debacles, and flow situation. The combination of the tools gives the surveillance needed in the presence of real-time analytics, where even insignificant mistakes can have vast consequences.

Others are polyglot support- Programmers have the freedom of selecting languages like Python, Java, or Node.js as they can accomplish their work, and native resiliency. Failures are confined to specific functionalities, while Step Functions provide mechanisms for retries, exception handling, and rollbacks, thereby ensuring continuity and maintaining data integrity. The architecture in general comprises a shift from the infrastructure-based to logic-based architecture. Scalable, cost-efficient pipeline provisions with event-driven triggers, ephemeral execution, and orchestration support real-time analytics. With this background in place, the second part now moves on to a discussion of how AWS Lambda is an efficient processing engine for these pipelines.

To further highlight the distinctions between serverless data engineering value propositions and conventional infrastructure-intensive architectures, it is essential to compare them in terms of elasticity, cost-effectiveness, fault tolerance, and developer experience. Such contrasts have been presented in the following table; it has specifically emphasized how the serverless models have become the foundation of the real-time analytics of today.

Table 1: Comparison of Traditional vs. Serverless Data Engineering Architectures

| **Feature** | **Traditional (Cluster-Based)** | **Serverless (AWS Lambda & Step Functions)** |
|---|---|---|
| **Scalability** | Requires manual scaling of clusters; often over- or under-provisioned | Automatic, event-driven scaling per workload |
| **Resource Utilization** | Idle resources consume costs even during low activity | Compute allocated only during execution |
| **Cost Model** | Fixed cost for infrastructure, regardless of usage | Pay-per-execution, highly cost-efficient for spiky workloads |
| **Fault Tolerance** | Failures in one node can affect the cluster | Isolated failures at the function level with retries & rollbacks |
| **Developer Focus** | Infrastructure-heavy; requires cluster management | Logic-centric; infrastructure abstracted away |
| **Latency** | Dependent on cluster readiness and resource limits | Millisecond-level responsiveness via function triggers |

Received: August 09, 2025

Volume 38 No. 7s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

This contrast can explain why serverless models are highly suitable in event-driven, real-time analytics and prime the application of AWS Lambda, which is the processing engine.

**4. AWS Lambda for Real-Time Analytics**

The processing engine should be considered to connect real-time analytics by which must be referred to as the logical extension of the architectural heritage of the serverless approach to data engineering, AWS Lambda as shown in Figure 1. This variable of its on-demand implementation makes it ideally applicable to circumstances when it is necessary to be responsive within a short term of time and without the additional expenditures on support of machinery. The analysis of encoder-decoder isolation data logic and hardware management, by testing with minimal costs, domain URLs can increase their analytic capacity without any issues related to costly growth [23]. Practically, the theoretical actions that might trigger the Lambda functions are a continuing stream of data. Ensure that you can take a look at the following example, as the combination of Amazon Kinesis records processing in milliseconds over the combination of the three services, Amazon should be able to support sub-second analytics on, necessarily, a clickstream data gathered by IoT telemetry or financial transactions. It is also possible to process log or sensor files near-real-time and make pipelines responsive without intentionally prepared clusters with the properties of the integration with Amazon S3 [24][25].

Heavily beneficial is the scaling up and down: an event will horizontally initiate its running environment, and thousands of operations can be carried out concurrently. NB, unlike in conventional clusters where the risk of under-utilization or overloading exists, in Lambda, the elasticities ensure balanced membership performance because the workloads vary [26]. Tasks like filtering, enrichment, and aggregation could be effectively processed through the use of Lambda, hence streamlining data pipelines. As an example, IoT reading can be denoted with location tags or aggregated with time before storing, and a single task modularized into a function to guarantee maintainability [27]. Lambda, however, imposes certain design restrictions. Time and memory constraints limit runtime, requiring complex analytical tasks to be partitioned or offloaded to services such as AWS Glue or EMR. On slower systems, every power-on sequence can introduce idle latency at startup, creating a critical bottleneck in the pipeline. The most common ways of reducing these effects are through the strategies of provisioned concurrency or planned invocations strategies [28]. Other uses of lambda include in real-time machine learning inference, during which small models can guess or score events in real-time, such as a recommendation generator designed to construct one-of-a-kind demand during e-commerce. Although these applications are constrained by resource capacities, their gradual advancement can be realized through the optimization of application frameworks [29].

Finally, monitoring and error handling are crucial. CloudWatch provides visibility into performance, while dead-letter queues capture failed events that can be reprocessed to ensure reliable and complete analytics outputs [30]. In general, AWS Lambda is the driver of serverless pipelines that are scalable and modular. The boundary of execution and a cold start is a highly structured work, but with the Elasticity of the Lambda, it is a core feature of real-time analytics

Received: August 09, 2025 

Volume 38 No. 7s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

today. The next section examines the AWS Step Functions, where it supplies the orchestration to integrate the Lambda tasks into end-to-end workflows that are hardy.

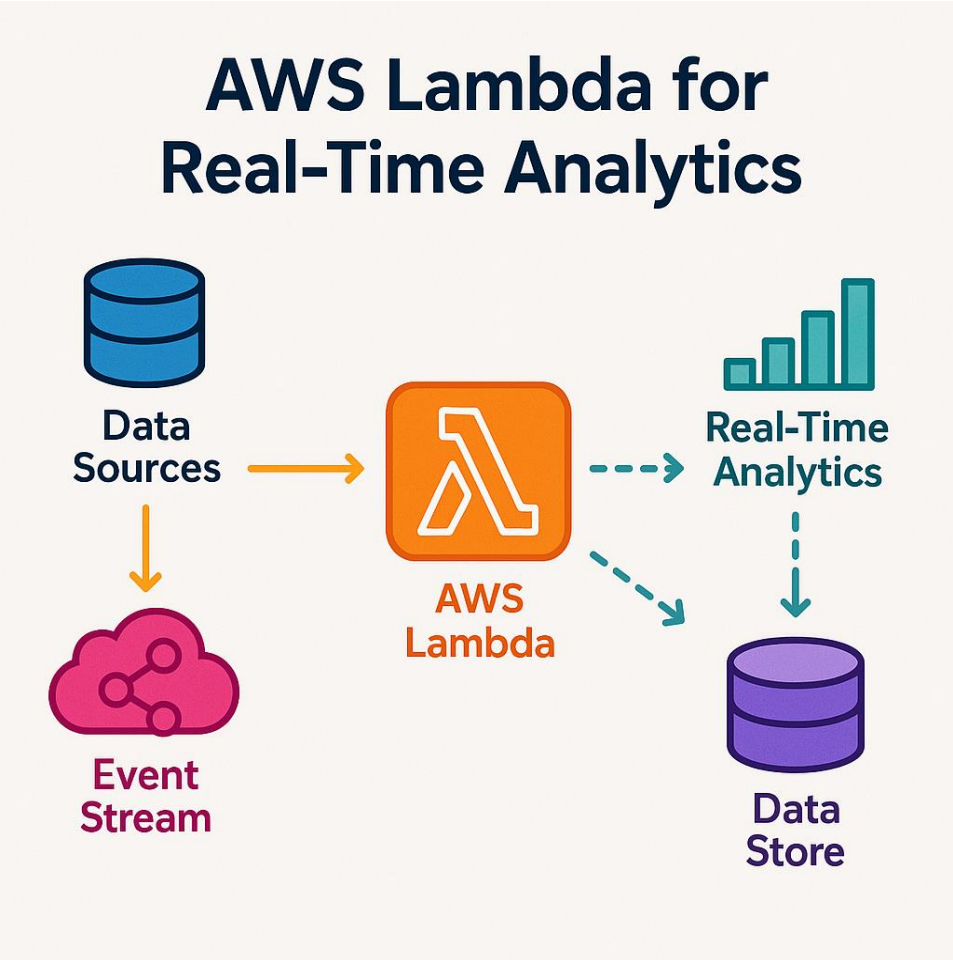


**Figure 1:** Real-Time Analytics with AWS Lambda. This diagram illustrates how data flows from various sources through event streams into AWS Lambda, enabling real-time processing and analytics before being stored in a data repository.

**5. Step Functions for Workflow Orchestration**

After exploring literature on Lambda as the processing engine of serverless pipelines, it becomes evident that real-time analytics extends beyond processing events in isolation, encompassing a broader and more integrated approach. The workflows of ingestion, validation, enrichment, storage, and alerting usually exist and are to be coordinated. AWS Step Functions provide such an orchestration layer and integrate the Lambda functions and other services in AWS into robust and coherent workflows [23].

The state management is the greatest strength of Step Functions. In Lambda, the entry point lambda is stateless, but Step Functions is a state model in which the output of one task is passed as input to another. This aids in the sequential, parallel, and branch workflows without the overhead of bringing orchestration logic into code that is difficult to maintain and obscure, and often in large-scale pipelines [24]. Step Functions are easy to pick up and to redo in noisy data sets. Custom lambda functions have been used to provide workflows with exponential backoff,

Received: August 09, 2025

Volume 38 No. 7s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

error catching, and rollback without implementing a bespoke retry logic in their applications. This will ensure consistency and a lifeline even when the situation is not usual [25].

An additional advantage is the ability to branch out workflows. As displayed in one of the instances of fraud detection, the legitimate flows may proceed to reporting storage, the dubious to machine learning scoring, and notifications. Such logic can either be expressed in a declarative manner, making the paths of the events visible [26]. Step Functions also integrate with a wide range of AWS services, including S3, DynamoDB, Glue, SageMaker, ECS, and SNS, without the need to pass through Lambda. Workflow or data into Redshift can be bypassed to make predictions, reducing glue code or reducing architecture complexity [27].

The art of observability can be enhanced through the use of execution histories and visual flow diagrams that allow the engineers to follow data inside the pipeline. This visibility, along with CloudWatch, can be utilized to outline areas of latency bottlenecks and can take several years to troubleshoot failures quickly [28]. Nonetheless, certain trade-offs must be acknowledged. Even small state changes can result in significant costs, while handling very large traffic volumes may lead to service-specific performance penalties. Minor latency overhead may be introduced by the orchestration layer, which is also relevant to an ultra-low-latency environment, e.g., trading. This can be reduced through using Step Functions and streaming systems like Kinesis [29].

Step Functions will be needed irrespective of those limits, since pipelines covering resilience, governance, and maintainability will charge their premiums, and at least Step Functions will be impacted. They have declarative workflows that are easier to trace and compliant, and are potentially relevant in regulated sectors (e.g., health care or finance) specifically [30]. Briefly, Step Functions are the orchestration instances that have been exploited to pick individual Lambda functions and make them entire real-time chains of analytics. They reconfigure the rhythm of less reactive implementation to formalized enterprise-wide processes by maintaining state, retries, utilizing branching, and integrating facilities. Since their existence is the task at hand, the next part yields to the universals of the difficulty and shortcomings of the adoption of those technologies.

## 6. Challenges and Limitations

Once a good source of real-time analytics, adoption of AWS Lambda and Step Functions faces the complexity of lacking drawbacks. Lambda processes the events stateless, and Step Functions are used to coordinate workflows, yet the reality is that there are several constraints that cloud exists and should be addressed through the sampling process to ensure their performance and reliability, and cost effectiveness [23].

Execution limits are another important consideration. Lambda functions are constrained by time, memory, and payload size, which makes them well-suited for lightweight operations but less effective for complex computations or broader generalizations. Step Functions address these trade-offs by enabling the chaining of smaller functions, which helps mitigate latency issues and reduce orchestration costs [24]. Cold starts are another important consideration. Functions that are not pre-optimized require an initialization period, resulting in delays that depend on their configuration. Such delays can be detrimental to low-latency sensitive

Received: August 09, 2025 

Volume 38 No. 7s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

workloads such as fraud detection. Provisioned concurrency can be used to curb this, but at an extra cost because some of the overheads of infrastructure can again be seen [25].

State management introduces additional complexity. Although the stateless condition of Lambda allows it to scale, the tasks that need to be persistent would become ideally more difficult to manage. Step Functions provide a state in the workflow, but the intermediate states at any moment tend to be so abundant that DynamoDB, S3, or RDS is required, providing downgrading dependencies sets and facility points [26]. Monitoring and debugging also pose challenges. The conventional servers are more elaborate, and their traceability and records are small; in contrast, safety from infrastructure is not observed in the serverless contexts. Despite the fact that observability is provided in CloudWatch and X-Ray, it is difficult to track momentary errors in the high-throughput pipelines [27].

Lock-back is also possible by vendors: as seen in AWS development, both Lambda and step functions have been well integrated with other services in the system, making disconnecting to other clouds simple and limiting. This dependency means reckoning may hinder migration whenever there are changes, which are strategic or regulatory in nature [28]. The cost effectiveness particularity which has already been mentioned as a positive factor, may become a negative one. The lambda costs are a product of the time of execution and the memory of execution, and the Steps Functions are a charge per state transition. Reboiler expenses, dead-letter collection, and interim storage have been costly in central high-frequency pipelines and can accumulate quickly unless the costs are masked discreetly [29].

Latency overhead is another trade-off. Step Functions adds a hierarchy of latencies to every state change. This may be appropriate in most common analytics applications, but other low-latency applications, such as trading and mission-critical Internet of Things, can be sensitive to such high-diaper low latencies, often running hybrid architectures that can mix serverless with unreliable high-low-latency streaming systems [30]. Finally, compliance and governance present significant challenges. While IAM and audit trails provide some support, the distributed nature of dynamically scaling infrastructures requires robust compliance frameworks. Methods and techniques must ensure data residency, encryption, and provenance factors that are especially critical for regulated sectors such as finance and healthcare [23]. All in all, even though both Lambda and Step Functions can be successfully utilized to offer flexible and scalable analytics, their implementation in environments requires users to have knowledge of the boundaries of execution time, cold start time lapse, state management, observability gaps, vendor lock-in, costs, latency, and compliance. These are not underestimated and as such must be sensitive architecture planning, tandem models, and constant simplification. The next section is going to turn to concrete use cases and applications where serverless pipelines nonetheless offer quantifiable benefits even on the grounds of these weaknesses in practice.

## 7. Use Cases and Applications

However, it has been found that AWS Lambda and Step Functions can be helpful in such sectors, where the analytics of the greenfields is required in real-time. They are readily scalable and economical enough so they can be very useful throughout finance to healthcare, to IoT areas [23]. Among the common applications is the identification of fraud when it comes to the utilization of money. Data on high velocity payments needs assessment at the level of

Received: August 09, 2025

Volume 38 No. 7s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

milliseconds. Lambda applies rules or other lightweight paradigms to real-time signals, by contrast to Step Functions, using signal events to be multiplexed in order to preview or alarm. This makes it possible to easily redress fraud without possessing a costly infrastructure [24].

IoT telemetry and predictive maintenance also benefit. Border equipment and border devices frequently emit sensor data that requires processing sequentially. Anomalies and expressions of Lamba filters can be aggregated to trigger a maintenance action step. Functions decrease the downtime and improve reliability [25]. In e-commerce, Lambda is used to power recommendation engines by processing clickstreams as customers navigate. Step Functions complement this by enabling enrichment and inference, allowing sessions to be personalized and dynamically stimulated with suggestions that enhance engagement and drive conversion [26]. In health care, Lambda is applicable during the live monitoring of vitals of a patient. Step Functions govern workflow, clinician alerts, dashboard, or predictive modeling. They are trackable to make sure that rigid healthcare standards are followed [27]. Logging Logs, Firewall, and authentication system logs gathered in the realm of cybersecurity can be processed by Lambda to identify a threat right away, yet Step Functions can coordinate multi-stage processing in order to escalate notices, or in order to ban the calling entity. This consideration is a multivariate sensitivity that promotes security against attacks [28].

And, finally, yet not least, there are real-time ETL processes. Filters and processes the incoming records, then loads them into S3 or loads them into Redshift. Step Functions pertain to orchestration of validation and notifications, which means that organisations can replace archaic ETL systems with skinny pipelines that are serverless [29]. The Lambda and Step Functions are fast, responsive, and shared processes in all of these, and reduce infrastructure overheads. Real sequential limitations, such as execution constraints or cost, are accounted for in prudent design, but real values in implementations are offered by pipelines that are serverless pipelines. The existence of the following applications is also reflected in the second section on what will come next in analytics in the context of serverless real-time [30].

To put the applications listed above into perspective, it is useful to draw industries versus their key analytics goals and the sort of value delivered by the serverless architectures specifically. This indicates that Lambda and Step Functions can be remapped across numerous domains without re-entangling the specific details of the underlying use cases.

**Table 2: Industry Applications of Serverless Real-Time Analytics**

| Industry | Analytics Objective | Serverless Benefit |
|---|---|---|
| Finance | Fraud detection, compliance monitoring | Sub-second detection, elastic scaling for spikes |
| IoT/Manufacturing | Predictive maintenance, anomaly detection | Continuous telemetry ingestion, reduced downtime |
| E-commerce | Real-time personalization, recommendations | Instant clickstream processing, modular workflows |

Received: August 09, 2025

Volume 38 No. 7s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

| | | |
|---|---|---|
| Healthcare | Patient monitoring, clinical alerts | Compliance-ready traceability, real-time alerts |
| Cybersecurity | Threat detection, log analysis | Multi-stage workflows for rapid incident response |
| Retail/ETL | Data cleansing and enrichment | Pay-per-use ETL, no cluster maintenance needed |

Industry-to-industry mappings suggest that, similar to use cases and benefit attributes, qualities such as scalability, cost efficiency, and resilience remain consistent across domains. Moving beyond real-life examples, the second part of the paper shifts focus to the future of work, highlighting the potential expansion of technology-mediated tasks and the broader applicability of serverless analytics.

## 8. Future Directions and Conclusion

The article on AWS Lambda and Step Functions in real-time analytics demonstrates that this powerhouse code is radical in data engineering regarding movers of serverless computing. The challenge of managing increased quantities of data, as well as agility and scaling to the services provided by industries, will certainly keep on increasing to a more significant-sized serverless model, driven not only by service improvement but also by additional integration with an increasing number of new technologies. The use of machine learning can be regarded as one of the enormous prospects of the future. Predictable compression of future models, compatibility with GPUs, and related services such as SageMaker are likely to allow inference complexity in real-time applications in emerging applications in the area of personalization, driven by anomaly detection and predictive maintenance. The other avenue is the appearance of the hybrid architectures. Persistent microservice or streaming engines, in their turn, are suitable for compute-intensive tasks/Worker or in terms of elasticity, to the benefit of which Lambda and Step Functions are characterized by a high scaling capacity. These combination solutions will provide a trade-off between prediction and levels of performance, especially in financial and autonomic systems.

There is also a priority given to optimizing costs and being sustainable. The micro cost enforcement and enhanced orchestration language has a chance of not merely ensuring pipelines are more cost-effective, but also more energy efficient, as part of the world's sustainability agenda in cloud computing. At the same time, thread governance will be enhanced and compliance enhanced. The serverless environments will result in more stringent privacy controls, and, in the short term, an industry like healthcare and finance will be more confident in their capacity to implement it. Lastly, in a serverless environment, one will sustain the process of entering the world of edge computing. IoT, robotic cars, and smart city extrasolar edge pipelines, as some will help to provide applications that are sensitive to latency through processing the data that is nearest to its origin. Lastly, the real-time analytics in elasticity and modularity and with less infrastructure management, can be efficiently executed on AWS Lambda and Step Functions. With some issues with the boundaries of execution, latency, and governance, developments in hybrid and compliance software and ecosystem significantly

Received: August 09, 2025

Volume 38 No. 7s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

imply that the near future does not lack serverless at the edge of data engineering. The next generation of real-time analytics will bring about the convergence of serverless, machine learning, and edge computing.

## References

[1] Adzic, G., & Chatley, R. (2017, August). Serverless computing: economic and architectural impact. In Proceedings of the 2017 11th joint meeting on foundations of software engineering (pp. 884-889).

[2] 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). Singapore: Springer Singapore.

[3] Jangda, A., Pinckney, D., Brun, Y., & Guha, A. (2019). Formal foundations of serverless computing. Proceedings of the ACM on Programming Languages, 3(OOPSLA), 1-26.

[4] Kim, Y., & Lin, J. (2018, July). Serverless data analytics with Flint. In 2018, IEEE 11th International Conference on Cloud Computing (CLOUD) (pp. 451-455). IEEE.

[5] Nastic, S., Rausch, T., Scekic, O., Dustdar, S., Gusev, M., Koteska, B., ... & Prodan, R. (2017). A serverless real-time data analytics platform for edge computing. IEEE Internet Computing, 21(4), 64-71.

[6] Joosen, A., Hassan, A., Asenov, M., Singh, R., Darlow, L., Wang, J., & Barker, A. (2023, October). How does it function? characterizing long-term trends in production serverless workloads. In Proceedings of the 2023 ACM Symposium on Cloud Computing (pp. 443-458).

[7] Shojaee Rad, Z., & Ghobaei-Arani, M. (2024). Data pipeline approaches in serverless computing: a taxonomy, review, and research trends. Journal of Big Data, 11(1), 82.

[8] McGrath, G., & Brenner, P. R. (2017, June). Serverless computing: Design, implementation, and performance. In 2017, IEEE 37th International Conference on Distributed Computing Systems Workshops (ICDCSW) (pp. 405-410). IEEE.

[9] Wen, J., Chen, Z., Jin, X., & Liu, X. (2023). Rise of the planet of serverless computing: A systematic review. ACM Transactions on Software Engineering and Methodology, 32(5), 1-61.

[10] Lannurien, V., D'orazio, L., Barais, O., & Boukhobza, J. (2023). Serverless cloud computing: State of the art and challenges. Serverless Computing: Principles and Paradigms, 275-316.

[11] Bermbach, D., Chandra, A., Krintz, C., Gokhale, A., Slominski, A., Thamsen, L., ... & Wolski, R. (2021, October). On the future of cloud engineering. In 2021, IEEE International Conference on Cloud Engineering (IC2E) (pp. 264-275). IEEE.

[12] Bebortta, S., Das, S. K., Kandpal, M., Barik, R. K., & Dubey, H. (2020). Geospatial serverless computing: Architectures, tools, and future directions. ISPRS International Journal of Geo-Information, 9(5), 311.

Received: August 09, 2025

Volume 38 No. 7s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

[13] Su, Y., Feng, D., Hua, Y., & Shi, Z. (2019). Understanding the latency distribution of cloud object storage systems. Journal of Parallel and Distributed Computing, 128, 71-83.

[14] Malawski, M., Gajek, A., Zima, A., Balis, B., & Figiela, K. (2020). Serverless execution of scientific workflows: Experiments with hyperflow, AWS Lambda and Google Cloud Functions. Future Generation Computer Systems, 110, 502-514.

[15] Osipowicz, K., Profyris, C., Mackenzie, A., Nicholas, P., Rudder, P., Taylor, H. M., ... & Doyen, S. (2023). Real-world demonstration of hand motor mapping using the structural connectivity atlas. Clinical neurology and neurosurgery, 228, 107679.

[16] Shafiei, H., Khonsari, A., & Mousavi, P. (2022). Serverless computing: a survey of opportunities, challenges, and applications. ACM Computing Surveys, 54(11s), 1-32.

[17] Christidis, A., Moschoyiannis, S., Hsu, C. H., & Davies, R. (2020). Enabling serverless deployment of large-scale AI workloads. IEEE Access, 8, 70150-70161.

[18] Taibi, D., El Ioini, N., Pahl, C., & Niederkofler, J. R. S. (2020). Patterns for serverless functions (function-as-a-service): A multivocal literature review. In International Conference on Cloud Computing and Services Science (pp. 181-192). Science and Technology Publications (SciTePress).

[19] Van Eyk, E., Toader, L., Talluri, S., Versluis, L., Uță, A., & Iosup, A. (2018). Serverless is more: From PaaS to present cloud computing. IEEE Internet Computing, 22(5), 8-17.

[20] Mampage, A., Karunasekera, S., & Buyya, R. (2022). A holistic view on resource management in serverless computing environments: Taxonomy and future directions. ACM Computing Surveys (CSUR), 54(11s), 1-36.

[21] Kounev, S., Abad, C., Foster, I., Herbst, N., Iosup, A., Al-Kiswany, S., ... & Yussupov, V. (2021). Toward a definition for serverless computing.

[22] Ogundipe, A. O. Advancing US Leadership in Cloud-Based Infrastructure and Serverless Architecture for Real-Time Data Analytics.

[23] Barrak, A., Petrillo, F., & Jaafar, F. (2022). Serverless on machine learning: A systematic mapping study. IEEE Access, 10, 99337-99352.

[24] Varghese, B., & Buyya, R. (2018). Next generation cloud computing: New trends and research directions. Future generation computer systems, 79, 849-861.

[25] Tusa, F., Clayman, S., Buzachis, A., & Fazio, M. (2024). Microservices and serverless functions—lifecycle, performance, and resource utilisation of edge-based real-time IoT analytics. Future Generation Computer Systems, 155, 204-218.

[26] Jarachanthan, J., Chen, L., Xu, F., & Li, B. (2022). Astrea: Auto-serverless analytics towards cost-efficiency and QoS-awareness. IEEE Transactions on Parallel and Distributed Systems, 33(12), 3833-3849.

[27] Cheng, B., Fuerst, J., Solmaz, G., & Sanada, T. (2019, July). Fog function: Serverless fog computing for data-intensive iot services. In 2019, IEEE International Conference on Services Computing (SCC) (pp. 28-35). IEEE.

Received: August 09, 2025 

Volume 38 No. 7s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

[28] Ansari, M., Ali, S. A., & Alam, M. (2022). Internet of Things (IoT) fusion with cloud computing: current research and future direction. International Journal of Advanced Technology and Engineering Exploration, 9(97), 1812.

[29] Tilles, J. (2020). Serverless computing on constrained edge devices.

[30] Dai, F., Hossain, M. A., & Wang, Y. (2025). State of the art in parallel and distributed systems: Emerging trends and challenges. Electronics, 14(4), 677.

Received: August 09, 2025 