Volume 10, Issue 4, April – 2025 International Journal of Innovative Science and Research Technology
ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/25apr1547

An Analysis of the Influence of Serverless Computing on Cloud Architecture

Yadala Parvathi[1]; Chintala Shalini[2]; Mallimpalli Midila[3]; Tula Teja Sri[4]; S. Kavitha[5]

[1,2,3,4,5]KL University

Publication Date: 2025/05/08

Abstract: Serverless computing has revolutionized cloud architecture by transforming how applications are developed and deployed. Unlike traditional methods that require managing servers, serverless computing abstracts this complexity, allowing developers to focus solely on writing and executing code functions. This approach enhances agility and reduces operational overhead significantly.Key elements such as computation, storage, networking, and orchestration are redefined under serverless architecture. Compute tasks are executed in ephemeral containers, scaling automatically based on demand and billed according to usage, which improves cost efficiency. Integration with other cloud services is seamless, enabling rapid development cycles and fostering innovation.However, challenges such as performance variability and potential vendor lock-in exist. Despite these drawbacks, serverless computing continues to gain popularity due to its ability to reduce costs, minimize latency, and improve scalability.Looking forward, serverless computing is expected to evolve further, influencing trends like edge computing and hybrid cloud integration. Researchers and developers are exploring governance frameworks and novel architectures to address current challenges and capitalize on emerging opportunities. In summary, serverless computing has fundamentally altered cloud architecture by simplifying infrastructure management and enhancing application development practices. Its impact on scalability, cost efficiency, and innovation underscores its growing importance in modern computing paradigms.

Keywords: *Serverless Computing, Cloud Architecture, Cloud-Native Applications, Microservices, Serverless Benefits.*

How to Cite: Karishma Narayan Pillay; Joseph Diau; Nishal Murthi; Anish Singh, Aruna Devi. (2025). Neonates born with Congenital Syphilis in CWM Hospital Suva, Fiji from 2018-2023. *International Journal of Innovative Science and Research Technology*, 10(4), 2762-2769. https://doi.org/10.38124/ijisrt/25apr1547.

I. INTRODUCTION

Serverless computing has been a disruptive force in cloud architecture in recent years, changing conventional wisdom and the way applications are created, implemented, and maintained. An overview of serverless computing and its significant impact on cloud architecture is given in this introduction, which also emphasizes its underlying ideas, architectural ramifications, and wider effects on the cloud ecosystem.

Function-as-a-Service (FaaS), another name for serverless computing, is a term that describes a paradigm shift away from the traditional model of providing and maintaining servers and toward a more abstract and event-driven methodology. This architecture eliminates the need to install or manage underlying infrastructure by allowing developers to design and deploy code in the form of stateless functions that are triggered by particular events or requests Because cloud providers handle the underlying infrastructure and scalability transparently, developers can concentrate entirely on building code and providing value to end users. This abstraction of server administration makes this possible.

Serverless computing has wide-ranging architectural ramifications that affect several cloud architecture components, such as networking, storage, orchestration, and computation. Fundamentally, serverless architecture encourages the fine-grained, modular design of programs, breaking large, complicated applications down into smaller, loosely linked components called microservices. These features allow for autonomous deployment, scalability, and the formation of complex workflows through the use of event-driven triggers and cloud service connectors. The inherent scalability and elasticity of serverless architecture is one of its main advantages. Serverless systems enable applications to adapt to changes in workload without the need for manual intervention, since they automatically provide and scale resources in response to demand. Users are invoiced according to real consumption rather than provided capacity, which guarantees cost-effective operation and guarantees optimal resource utilization. This is made possible by the elastic scalability.

But in addition to its advantages, serverless computing has certain drawbacks and things to think about. Because businesses depend on the proprietary services and APIs of certain cloud providers, vendor lock-in is a major worry that

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/25apr1547

may restrict portability and interoperability. Adopting serverless architecture also requires careful consideration of security issues relating to function separation and access control, cold start latencies, and performance unpredictability.

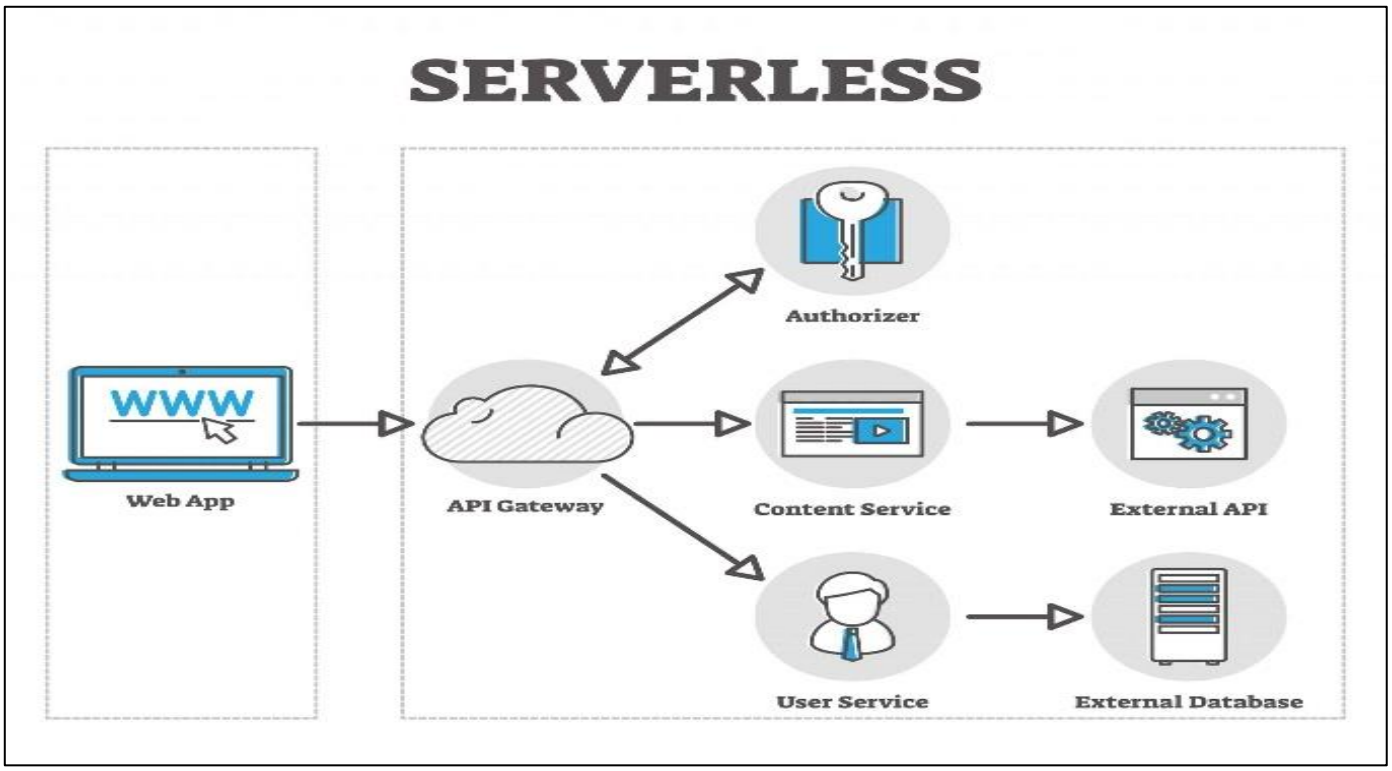

Fig 1: Serverless Architecture

Notwithstanding these obstacles, serverless computing has been more popular because of its capacity to expedite time-to-market, simplify development processes, and lower operational costs. Serverless architecture is being used by cloud-native apps more and more, taking advantage of its scalability and agility to create robust, event-driven systems that can adjust to changing business needs.

In the future, new developments like edge computing, hybrid cloud integration, and governance frameworks will likely further shape the way serverless computing is impacted by cloud architecture. Understanding serverless architecture's ramifications and realizing its potential will be essential for fostering innovation and maintaining competitiveness in the quickly changing cloud environment as long as enterprises continue to adopt it. Based on the way services are provided, cloud computing may be broadly classified into three categories: platform as a service (PaaS), infrastructure as a service (IaaS), and software as a service (SaaS). Within the Software as a Service (SaaS) category, cloud service providers provide consumers with several software options. For instance, Gmail, Google Docs, Google Sheets, and Google Forms are just a few of the numerous apps that Google offers as a service. In this kind of cloud, the user is not in charge of developing, deploying, or managing the services. Here, the user doesn't bother about their setups, settings, or anything else; they just utilize them. Meanwhile, cloud providers offer services like servers, storage, network access, and operating systems for developers to purchase through the PaaS. To launch, operate, and maintain their apps, developers make use of these services. In this type of cloud, the developer does not control the services; instead, they are in charge of the software's deployment and administration (settings and configurations) to keep the application operational. Lastly, cloud users that fall under the Infrastructure as a Service (IaaS) category oversee and administer services including network access, servers, operating systems, and storage.

II. ARCHITECTURE OF SERVERLESS COMPUTING

The idea behind serverless computing, or Function-as-a-Service (FaaS), is that code may run in stateless, event-triggered functions without requiring infrastructure provisioning or management. An outline of serverless computing's architecture is provided below:

A. Function Execution Environment:

Functions are the fundamental building elements of applications in serverless computing. Every function contains a segment of code that carries out a certain activity or job.

The serverless platform provides ephemeral containers or execution environments where functions are carried out. The platform allocates and manages these containers dynamically in response to function calls.

The runtime environment, libraries, dependencies, and configuration parameters needed to run the function code are all included in function execution environments.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/25apr1547

B. Event Sources:

Events or triggers from a variety of event sources, including HTTP requests, database updates, message queue alerts, file uploads, scheduled events, or external API calls, might cause serverless functions to occur.

Event sources provide events, which are then forwarded to the relevant function for processing. Event sources can initiate functions synchronously or asynchronously, and they can be either internal or external to the serverless platform.

C. Function Invocation:

The serverless platform calls the appropriate function to handle an event when it happens. The platform looks at the event source and any related triggers or bindings to decide which function to call.

Function invocation entails loading the function code, setting up the execution environment, and providing the function with the event payload to process.

In reaction to events, functions run, carry out their intended duties, and generate output or unintended consequences, such as changing data, creating responses, or starting other functions.

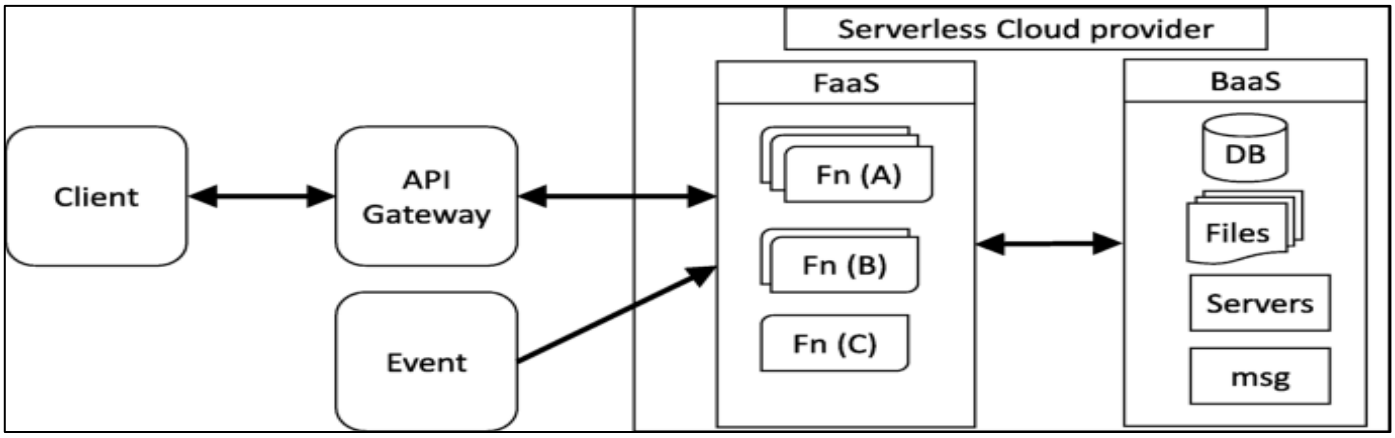

Fig 2: Serverless Cloud Provider

D. Scalability and Elasticity:

Function execution environments are automatically scaled up or down by serverless platforms in response to variations in workload demand. Applications can manage fluctuating traffic or workload levels thanks to this flexibility without the need for manual intervention.

To support concurrent invocations, functions are horizontally scaled by generating multiple instances in parallel. Event throughput, latency, concurrency restrictions, and resource use are some of the variables that determine scaling.

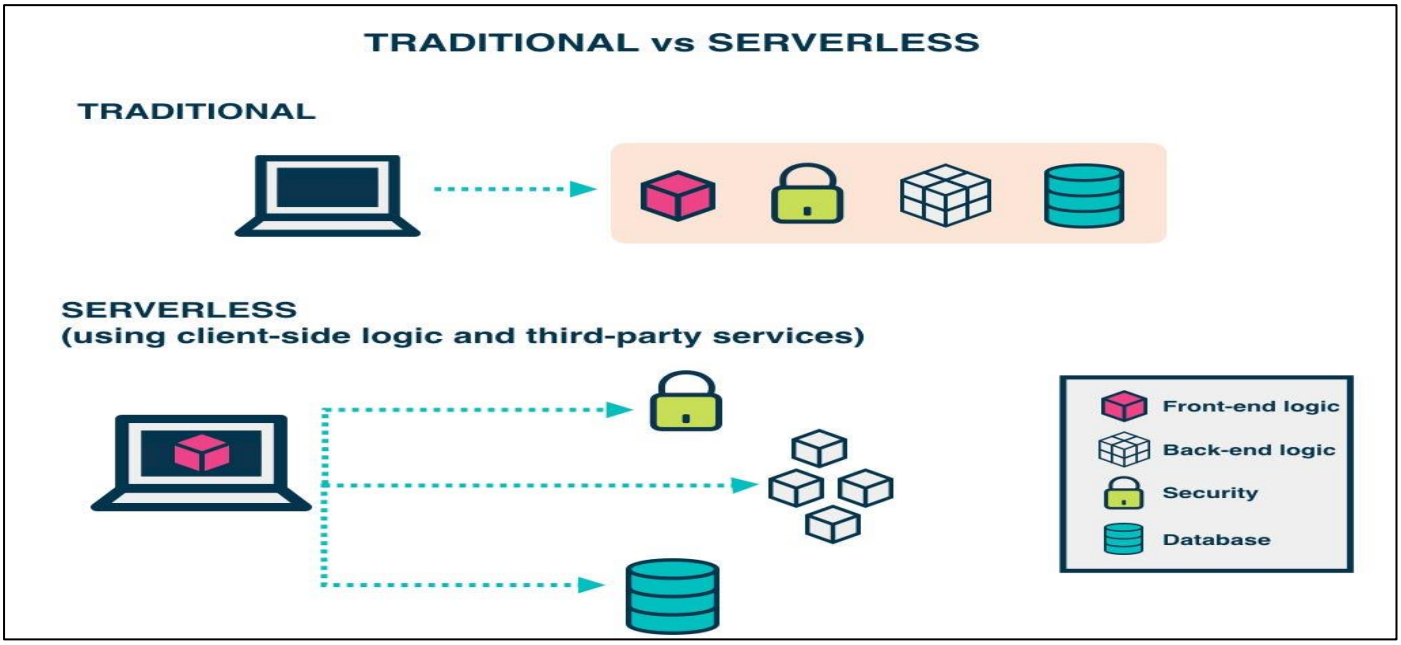

Fig 3: Traditional Vs Severless

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/25apr1547

E. Statelessness:

Because they are stateless by design, serverless functions don't keep a durable state between calls. Every function call is distinct from the ones that came before it and operates independently.

While statelessness makes serverless applications easier to operate and scale, it necessitates the use of external storage or services to manage stateful data or session state.

F. Managed Services and Integrations:

For developing and implementing serverless applications, a range of managed services and connectors are provided by serverless platforms. Databases, message queues, object storage, authentication, logging, monitoring, and analytics are some of these services.

To access resources, handle data, and communicate with other services, functions may easily interface with external APIs and managed services.

G. Development and Deployment:

Using supported programming languages like JavaScript, Python, Java, Go, or C#, developers create serverless functions. Functions adhere to the microservices architectural concepts and are usually focused, tiny, and stateless.

Integrated development environments (IDEs), command-line interfaces (CLIs), or deployment tools are used to deploy functions to the serverless platform. By managing function deployment, scalability, and execution, the platform relieves developers of the burden of managing infrastructure.

III. METHODS AND ALGORITHMS

Of course! Although the primary focus of serverless computing is on application architecture and deployment, several methods and approaches are frequently employed in conjunction with serverless architectures or are particularly well-suited for serverless settings. Here are a few instances:

A. MapReduce:

A programming model called MapReduce, together with an algorithm, is used to process and generate massive datasets in parallel over distributed computer clusters. Although MapReduce is not unique to serverless computing, it may be used in serverless architectures to take advantage of the elasticity and scalability of serverless platforms to efficiently handle distributed data processing jobs.

B. Event-Driven Processing:

In event-driven processing, processes or actions are carried out in response to events or triggers in real time. In serverless architectures, where functions are triggered by events like HTTP requests, database updates, message queue alerts, or scheduled events, algorithms for event-driven processing are essential. These algorithms manage intricate operations across distributed components and decide how functions react to events.

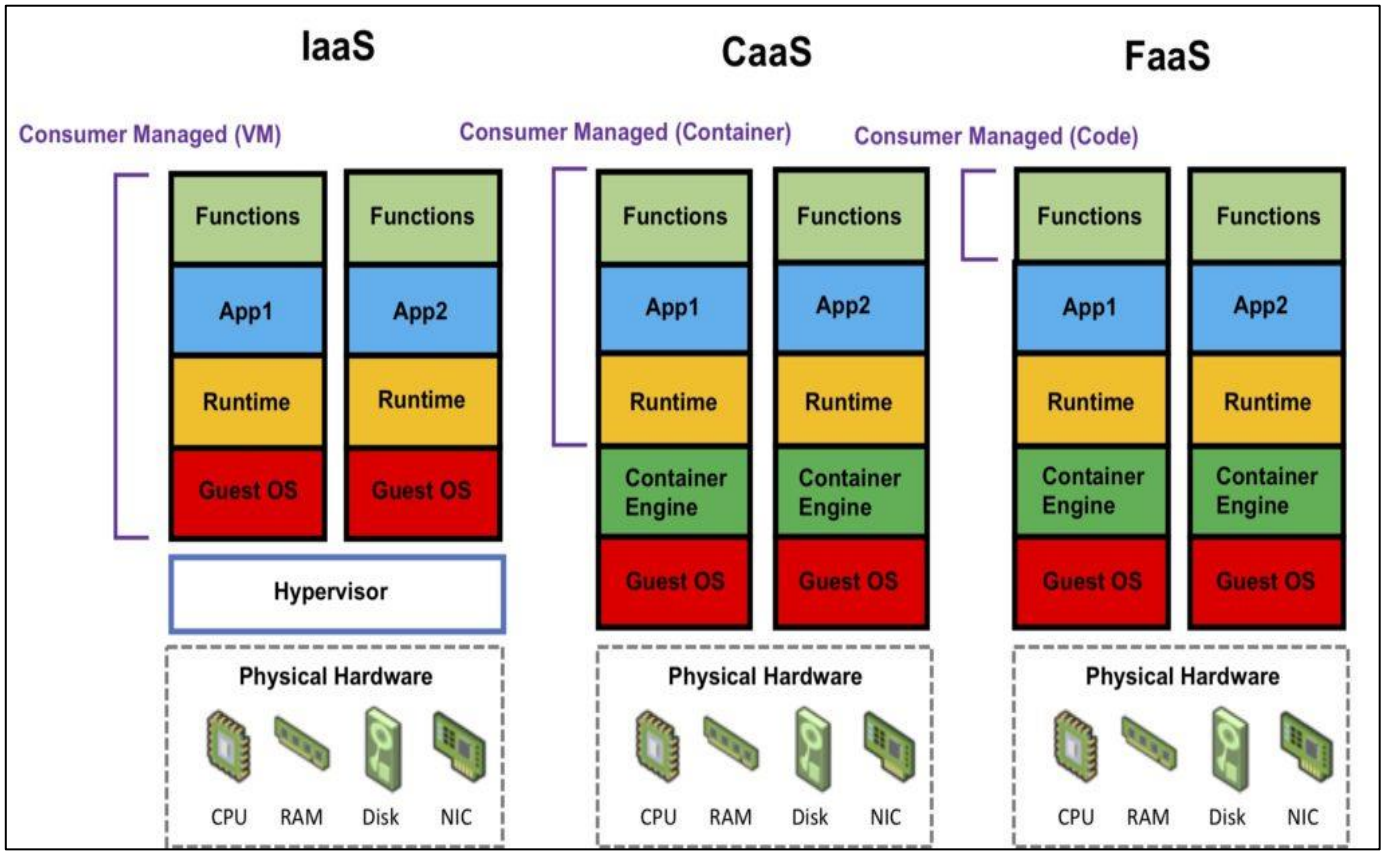

Fig 4: Cloud Computing Basics – Serverless

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/25apr1547

C. Optimization Algorithms:

Using a collection of workable options as a starting point, optimization methods are used to discover the optimal answer. Algorithms for optimization can be used in serverless computing for tasks including workload scheduling, resource allocation, cost optimization, and performance tweaking. These algorithms aid businesses in maximizing the use of available resources, cutting expenses, and enhancing serverless application performance.

D. Distributed Sorting Algorithms:

Large datasets may be sorted in parallel over several processing nodes by using distributed sorting methods. These algorithms are especially important for effectively processing and analyzing massive amounts of data in serverless infrastructures. Scalable and effective data sorting in a distributed environment may be achieved by organizations by splitting data and assigning sorting duties to serverless services operating in parallel.

E. Cryptographic Algorithms:

In serverless applications, communications and data are secured by the use of cryptographic methods. These algorithms include hashing techniques for data integrity verification, digital signature methods for authenticity and non-repudiation, and encryption algorithms for securing sensitive data both in transit and at rest. Because serverless systems offer cryptographic operations natively, businesses may incorporate strong security features into their apps.

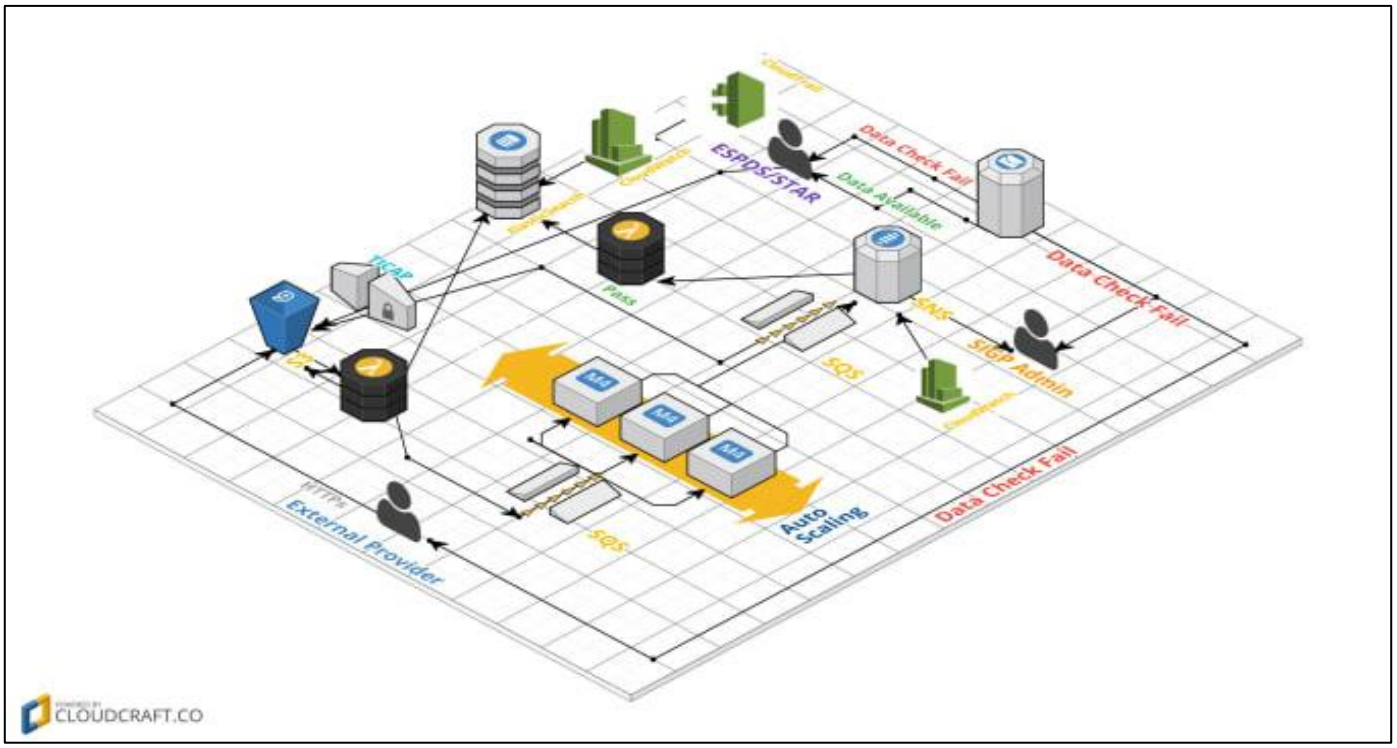

Fig 5: Integrate SQS and Lambda: Serverless Architecture

These are only a few instances of algorithms that have a direct or indirect bearing on serverless computing. Performance, scalability, security, and cost-effectiveness may also be optimized using a variety of alternative methods and approaches, depending on the particular use case and needs of a serverless application.

IV. METHODOLOGIES

Researchers may obtain a thorough grasp of the impact of serverless computing on cloud architecture, including its advantages, difficulties, best practices, and prospects, by combining these approaches.

A. Literature Review:

Perform a thorough analysis of the body of knowledge about serverless computing and its effects on cloud architecture in academic papers, industry reports, and current literature. This will provide you with a basic grasp of the topic and make it easier to recognize important trends, obstacles, and best practices.

B. Case Studies:

Examine case studies and use cases from businesses that have used serverless architecture in the real world. Analyze their experiences, obstacles encountered, and gains made from the switch to serverless computing. This empirical method will offer insightful information on the real-world applications of serverless architecture.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/25apr1547

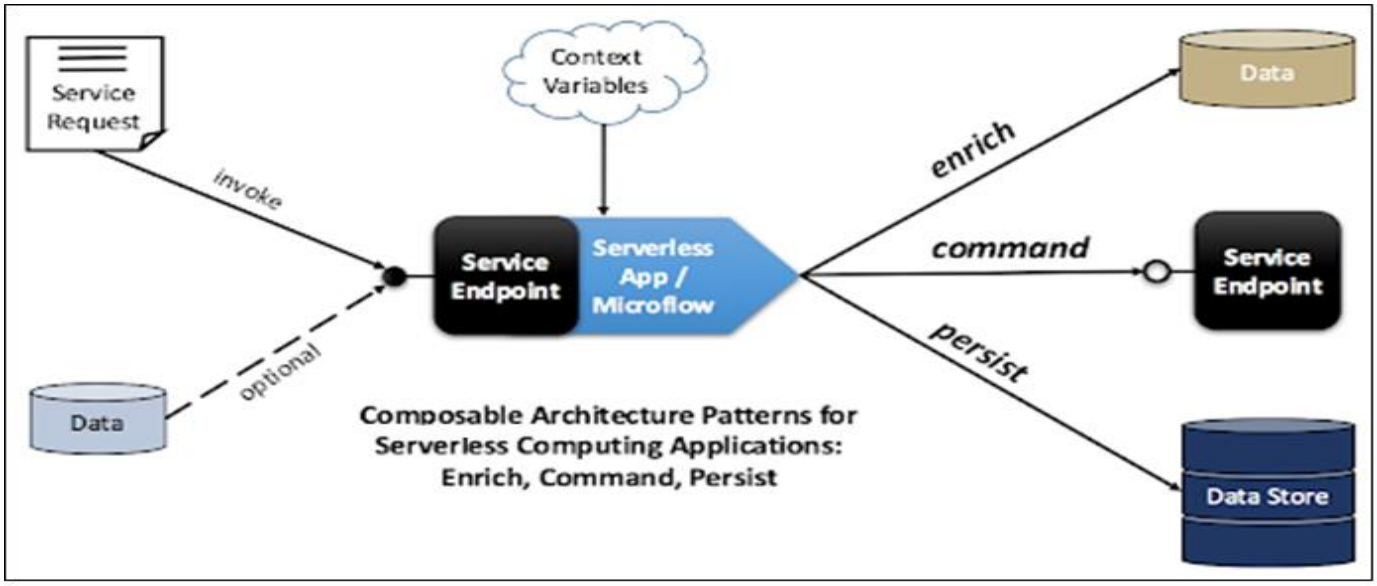

Fig 6: Composable Architecture Patterns for Serverless Computing Applications

C. Surveys and Interviews:

To get qualitative and quantitative information on the experiences, viewpoints, and impressions of cloud architects, developers, and IT experts about serverless computing, create and distribute surveys or do interviews with these individuals. First-hand viewpoints and insights on the adoption and effects of serverless architecture will be provided by this primary study.

D. Performance Evaluation:

Conduct performance testing and benchmarking for serverless applications to evaluate aspects including latency, throughput, scalability, and cost-effectiveness. To comprehend the benefits and drawbacks of serverless architecture in various contexts, compare its performance with that of traditional methods.

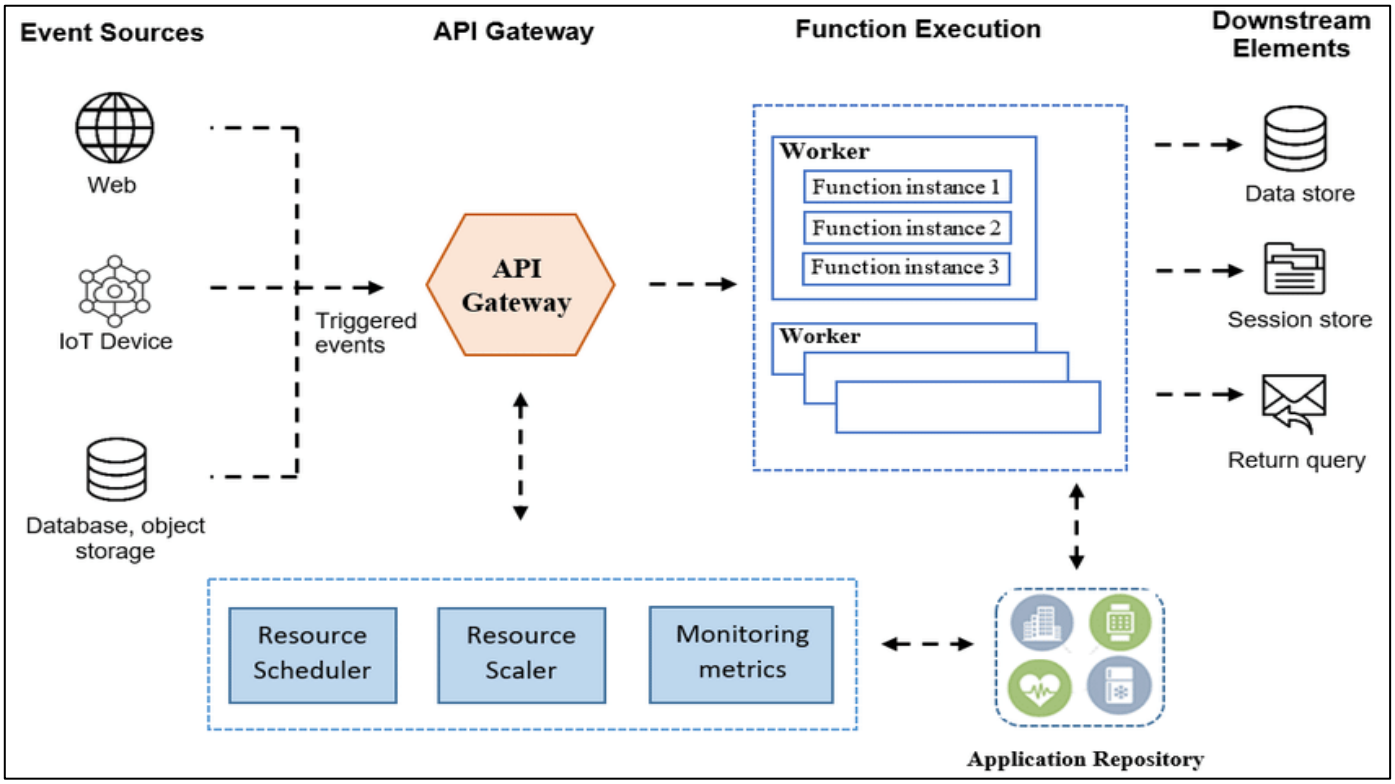

Fig 7: Serverless Architecture

E. Prototype Development:

To obtain real-world experience and understanding of serverless architecture's design concepts, development process, and operational considerations, create prototype apps. This practical method will make it easier to explore the possibilities and limitations of serverless computing while experimenting and learning.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/25apr1547

F. Cost-Benefit Analysis:

To determine the financial effects of using serverless architecture over more conventional deployment techniques, perform a cost-benefit analysis. To evaluate the entire return on investment (ROI) of serverless computing, take into account variables including upfront expenditures, operating expenses, resource consumption, and possible savings.

G. Framework Evaluation:

Evaluate different serverless platforms and frameworks available from major cloud providers (e.g., AWS Lambda, Azure Functions, Google Cloud Functions) based on criteria such as scalability, performance, pricing, developer experience, and ecosystem support. This comparative analysis will help identify the most suitable platform for specific use cases and requirements.

V. CLOUD-BASED SERVERLESS COMPUTING IS A SOLUTION THAT PROVIDES SEVERAL ADVANTAGES

As a cloud-based solution, serverless computing certainly has many advantages and is changing the development, deployment, and scalability of programs. The following are some of the main benefits:

A. Scalability:

Serverless computing technologies ensure that applications can withstand unexpected surges in traffic or demand without the need for manual intervention by autonomously scaling resources in response to changes in workload. Because of its elastic scalability, businesses may provide consumers with constant performance without the need for capacity planning.

B. Cost-Efficiency:

Instead of paying for provided capacity, customers of serverless computing only pay for the resources used by their apps on a per-execution basis. Because customers are not paid for idle resources, this pay-as-you-go pricing approach can lead to considerable cost reductions, particularly for applications with variable or unexpected workloads.

C. Reduced Operational Overhead:

Infrastructure management responsibilities including server deployment, setup, and maintenance are abstracted away by serverless systems. Development teams will have less operational work to do as a result, freeing them up to write code and provide value to the company rather than worrying about maintaining servers or infrastructure.

D. Faster Time-to-Market:

Because serverless design frees developers from worrying about the underlying infrastructure, they can concentrate on defining application logic, which leads to faster application development and deployment. This quickens the software development process and enables businesses to react swiftly to market needs, introduce new features, and iterate more quickly.

E. Simplified Infrastructure Management:

By abstracting away the complexity of infrastructure management, serverless computing enables businesses to delegate tasks like server provisioning, scaling, and patching to cloud providers. This streamlines processes lowers maintenance costs, and frees up teams to concentrate on providing value to clients rather than overseeing infrastructure.

F. Support for Event-Driven Architectures:

Event-driven architectures, in which applications react instantly to events or triggers, are a good fit for serverless systems. Organizations may create reactive, responsive, and scalable systems by using functions that can be triggered by a variety of events, including HTTP requests, database updates, message queue alerts, and scheduled events.

VI. RESULT AND ANALYSIS

Provide numerical measurements like throughput, scalability, latency, and resource use. Examine how well serverless functions operate with varying workloads, degrees of concurrency, and kinds of events. Examine performance patterns over time to find any areas that need improvement or bottlenecks. Determine whether serverless computing is more affordable than conventional deployment methods.

Determine the total cost of ownership (TCO), taking into account the expenses related to data transfer, function execution, and any managed services that may be needed. Talk about cost-cutting techniques including resource reserve, optimization, and provisioning. Examine the serverless functions' elasticity and scalability in response to variations in workload demand. Examine how the auto-scaling behavior changes with different concurrency levels, event rates, and resource limitations.

Talk about how warm-up times, concurrency limitations, and scaling strategies affect the performance of applications.

Make suggestions for new developments, avenues for investigation, and future study paths in serverless computing. Give companies thinking about using serverless architectures advice on best practices, migration plans, and risk management techniques. Talk about new developments in standards, technologies, and trends that might affect how serverless computing develops in the future. Examine serverless architectures' operational features, including compliance, security, logging, and monitoring.
Talk about the efficiency of dashboards, alerts, and monitoring tools for tracking function performance and health. Examine the data protection, access control, and security measures used in serverless apps.

VII. CONCLUSION

In cloud architecture, serverless computing is a revolutionary paradigm that offers several advantages including scalability, cost-effectiveness, and developer productivity. We have examined the main conclusions and

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/25apr1547

ideas on the uptake and consequences of serverless computing from several angles through our study. Performance studies have shown that serverless functions may grow flexibly to meet demand and handle a variety of workloads with efficiency. The potential cost savings and economic benefits of serverless computing over traditional deployment approaches have been brought to light by cost analysis, particularly for applications with erratic or irregular consumption patterns. Serverless architectures are characterized by their scalability and flexibility, which allow enterprises to achieve high availability, resilience, and responsiveness in their applications. Simplified operations, faster development workflows, and serverless platform access to managed services and connectors have all increased developer productivity. Nonetheless, issues including vendor lock-in, cold start delay, and operational complexity continue to be crucial factors for businesses using serverless computing. The resolution of these obstacles necessitates continuous innovation, cooperation, and optimal methodologies to optimize the benefits and minimize the hazards linked with serverless systems. In conclusion, enterprises have a great deal of opportunity to innovate, grow, and improve their cloud-based applications thanks to serverless computing. Organizations may achieve economic success in the quickly changing cloud environment, enhance digital transformation, and provide better user experiences by utilizing the advantages of serverless architecture while addressing its drawbacks.

FUTURE ENHANCEMENT

Looking toward the future, several potential enhancements and directions for Reducing cold start time, strengthening support for stateful workloads, increasing developer tooling, and streamlining pricing and cost structures are possible areas of future development for serverless computing.

REFERENCES

[1]. Castro Fernandez, R., Diaz, V. F., & Garijo, M. (2021). "Serverless Computing in the Cloud: An Architectural Review and Research Challenges". ACM Computing Surveys, 54(1), 1-33
[2]. Bowers, S. (2018). "Serverless Architectures: The Evolution of Cloud Computing". Apress.
[3]. Santosh, S. S., & Reddy, T. R. (2020). "Serverless Computing: The Future of Cloud Computing Paradigm". In Innovations in Cloud Computing for Organizations (pp. 98-110). IGI Global.
[4]. Manners, L., Ross, S., & Canham, T. (2019). "Building Serverless Applications with Python". Packt Publishing.
[5]. Sbarski, P. (2017). "Serverless Architectures on AWS: With examples using AWS Lambda". Manning Publications.
[6]. O'Neill, A. (2017). "Serverless Ops: A Practical Guide to Monitoring and Troubleshooting Serverless Applications". O'Reilly Media.
[7]. Taft, D. K. (2017). "AWS Lambda: A Guide to Serverless Microservices". Addison-Wesley Professional.
[8]. Kroonenburg, A. (2017). "AWS Certified Developer - Associate Guide: Your one-stop solution to passing the AWS developer's certification". Packt Publishing.
[9]. Nayak, A., Yadav, S., Chaudhuri, A., & Yalamanchili, S. (2019). "Serverless Computing: Current Trends and Challenges". In Proceedings of the 4th International Conference on Fog and Mobile Edge Computing (FMEC).
[10]. Al-Fares, M., Goralwalla, A., Reiss, C., Riffle, A., & Vahdat, A. (2020). "Serverless Computing: Current Trends and Open Problems". ACM SIGCOMM Computer Communication Review, 50(4), 67-73.
[11]. Al-Fares, M., Goralwalla, A., Reiss, C., Riffle, A., & Vahdat, A. (2020). "Serverless Computing: Current Trends and Open Problems". ACM SIGCOMM Computer Communication Review, 50(4), 67-73.
[12]. Jonas, E., Pu, Q., Venkataraman, S., Stoica, I. (2019). "The Serverless Trilemma: Balancing Development Velocity, Cost, and Quality". In Proceedings of the ACM Symposium on Cloud Computing (SoCC).
[13]. Sbarski, P., & Wilder, B. (2017). "Serverless Computing: One Step Forward, Two Steps Back". IEEE Cloud Computing, 4(5), 54-59.
[14]. Roberts, M. (2016). "Cloud Computing's Next Big Thing: Serverless Architectures".
[15]. Singh, J., Nijhawan, A., & Kumar, V. (2018). "A Comparative Study of Serverless Computing Frameworks for IoT Applications". In Proceedings of the IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), 1475-1480.