International Journal of Current Engineering and Technology
E-ISSN 2277 – 4106, P-ISSN 2347 – 5161

Available at http://inpressco.com/category/ijcet

Research Article

Serverless Database Architectures: Myths, Benefits, and Real-World Adoption Roadmap for Large Enterprises

Veeravenkata Maruthi Lakshmi Ganesh Nerella*

Sr. Database Administrator, Summerfield, NC, USA

Received 01 Aug 2025, Accepted 24 Aug 2025, Available online 25 Aug 2025, **Vol.15, No.4 (July/Aug 2025)**

Abstract

Cloud data management is being transformed by serverless database designs, which provide new ways to optimize resources and abstract infrastructure. These platforms enable seamless scalability and cost-efficiency through auto-scaling mechanisms and pay-per-use pricing models, aligning operational costs with real-time demand. This paper provides a comprehensive review of serverless database systems, tracing their evolution from traditional architectures to cloud-native paradigms. It includes a comparative analysis of leading serverless offerings like AWS Lambda, Azure Functions, and Google Cloud Functions evaluated across critical dimensions of performance, scalability, and cost. Furthermore, the study investigates real-world adoption across industries, including finance, retail, healthcare, and technology, illustrating how serverless databases support rapid development, improve resource utilization, and enhance system resilience. In addition to outlining key benefits, the paper addresses prevalent myths and operational challenges hindering adoption. A structured, enterprise-focused adoption roadmap is proposed to guide large organizations through the transition. Finally, future research directions are discussed to address current limitations and expand the capabilities of serverless database platforms in complex enterprise environments.

Keywords: *Serverless Computing, Database Architecture, Cloud-native, cost optimization, Enterprise Cloud Adoption.*

Introduction

The use of cloud computing has transformed the design, deployment, and scalability of contemporary applications in recent years. The abstraction of server administration by serverless architecture, which frees developers to concentrate only on creating and implementing application logic, is one of the paradigm's most revolutionary developments[1]. Serverless computing allows organizations to avoid over-provisioning and grow flexibly in response to varying workloads by providing backend services on a pay-as-you-go approach[2]. Despite the name "serverless," real servers are still in use, but the cloud provider is solely responsible for their provisioning, scalability, and upkeep[3]. As a result, organizations benefit from reduced operational complexity, improved scalability, and enhanced cost efficiency. Enterprises adopting serverless architectures report an average 47% reduction in operational overhead compared to traditional cloud models.

Serverless computing is not limited to application logic; it has also significantly impacted data management. Serverless databases represent a fundamental reimagining of traditional database systems[4]. These databases scale automatically based on demand, which does not require ease of capacity planning and which enables consumption-based pricing [5]. Analyses reveal that serverless databases could scale linearly up to 2,500 concurrent connections and still perform under 100 milliseconds, and thus are highly appropriate in workloads with unpredictable traffic patterns. it does its job.

Additionally, costs have been reduced by 3560% reported by enterprises in the move to serverless databases, particularly in workload environments with uncertain loads. Serverless databases' performance benchmarks show 99 percentile latencies under 200ms, regardless of rapid scale events with more than 1,000 concurrent users. These performance metrics reflect just how much serverless databases has come along in its march towards enterprise acceptance.

Scalability, agility and efficiency are vital in the case of large enterprises. Serverless architectures meet such needs by saving the management effort of infrastructure, a faster innovation cycle, and decreased time-to-market[6]. Serverless databases are gaining popularity in many industries, including banking, retail, healthcare, and AI-driven applications, due to

*Corresponding author's ORCID ID: 0009-0008-6149-846X
DOI: https://doi.org/10.14741/ijcet/v.15.4.8

their ability to accommodate several data models and interface with ecosystems that are native to the cloud[7]. The replacement of capacity optimization with resource allocation based on consumption allows large companies to directly match costs with use, maximize resource utilization and assign engineering skill to strategic goals, not infrastructure support[8]. Nevertheless, serverless adoption does not come without complications. Complexity of migration, cold start latency, security issues and lock-in are all costly impediments to overcome[9]. The strategies to address these issues are very delicate, considering the technical and organizational dynamics.

Structure of the paper

This paper is organized in the following way: Section II presents a comprehensive overview of serverless architectures. Section III examines prevalent myths, misconceptions, and the key challenges associated with serverless database systems. Section IV analyses the benefits and enterprise-level adoption trends of serverless solutions. Sections V and VI synthesize the existing literature and outline the conclusions along with potential directions for future research.

Understanding Serverless Architectures

The serverless architecture is based on the FaaS concept, which allows cloud platforms to run programs without requiring users to manage the underlying infrastructure. It is inherently stateless and fully managed by cloud service providers, who handle provisioning, scaling, and execution of user-defined functions[10]. The architecture is event-driven, and function-centric developers simply upload code written in any supported language, and the system automatically determines which function to invoke in response to specific events, like HTTP requests[11]. Upon receiving a request, the platform creates a container instance, routes the event to the correct function, and generates execution logs along with the final response. The function is terminated once execution completes. This model enables high scalability and cost-efficiency by allocating resources only when needed. Figure 1 illustrates the architecture of a distributed serverless system, highlighting how client requests are routed through a federation to backend processes for execution via load-balanced replicas and a database engine.

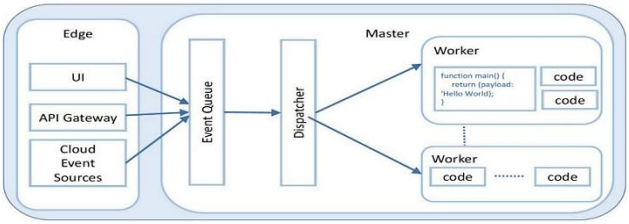

Fig.1 Serverless Architecture

Evolution of Serverless from Traditional Architectures

An evolution from standalone functionalities, serverless computing now enables full-stack development by combining authentication, data processing, and database management services with state-of-the-art frontend frameworks [12]. Container services that provide serverless features to containerized applications, such as AWS Fargate and Google Cloud Run, give developers more freedom without requiring them to manage infrastructure. These advancements reflect the growing maturity of the serverless ecosystem, enabling a broader range of complex, scalable applications. This evolution from basic to multi-component architectures is illustrated in Figure 4.

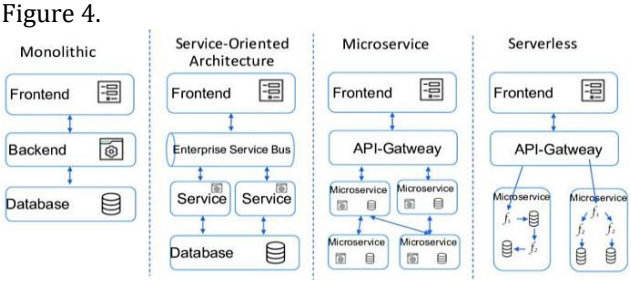

Fig.2 The Evolution of Software Architectures

Figure 2 compares Monolithic, SOA, Microservice, and Serverless architectures. Monolithic systems are single units requiring full redeployment for changes. SOA modularizes components via an ESB but adds complexity. Microservices enable independent services with separate databases and API-based communication. Serverless replaces persistent services with event-driven functions, offering scalability and cost efficiency, but it depends on cloud providers.

Serverless vs. Traditional & Cloud-native Databases

Serverless database architectures represent a paradigm shift in database management, offering fully managed services provided by cloud providers. These systems decouple the data storage layer from the compute layer, enabling independent scaling of each component based on fluctuating workloads [13]. This contrasts with traditional database deployments, where a dedicated server (physical or virtual) is provisioned and managed, requiring explicit resource allocation, configuration, and ongoing maintenance. In the serverless model, the cloud provider manages both the storage and compute resources, abstracting away the underlying infrastructure concerns and automating scaling operations. Even with the benefits of containerization and orchestration, cloud-native databases still need infrastructure monitoring[14]. Serverless databases, in contrast, employ a fully managed, event-driven model that closely aligns with dynamic and high-concurrency workloads, making them increasingly suitable for modern, real-time, and microservice-oriented applications[15]. This architectural distinction has a significant impact on

software development practices and data manipulation strategies, simplifying application deployment and optimizing resource utilization, as shown in Figure 3.

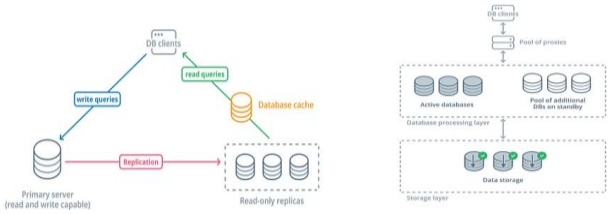

Fig.3 Traditional database vs Serverless database

Types of serverless databases: SQL vs NoSQL

Serverless databases can be classified into SQL and NoSQL types, each serving different application requirements.

SQL is used for querying and managing relational databases. It enables users to perform operations such as querying, inserting, updating, and deleting records using a predefined schema. SQL supports advanced features like transactions, joins, and stored procedures, which ensure data integrity and consistency[16]. Serverless relational databases like AWS Aurora Serverless offer the capabilities of relational databases with autoscaling and lower operational overhead.

NoSQL (Not Only SQL) databases based on NoSQL data models, that is, document store, key value pairs, wide column store, and graph databases. These are schema-less or schema-flexible databases that support -fast changes to data structure[17]. NoSQL systems are highly scalable, fault-tolerant and able to process unstructured or semi-structured data in an efficient manner. Applications requiring flexibility (and real-time responsiveness) may take advantage of serverless NoSQL databases like Azure Cosmos DB, Firebase (Firestore), and FaunaDB.

Comparison of SQL vs NoSQL

The differences between SQL and NoSQL databases are their structure, scalability, and purpose. SQL databases are schema-driven and well-suited to structured data and enterprise systems due to compliance with ACID and support of complex queries. NoSQL databases are schemaless and support horizontal scaling, and are better suited for unstructured data and real-time applications. A detailed comparison is presented in Table I:

Table 1 Comparison between SQL vs NoSQL

Feature	SQL	NoSQL
Full Form	Structured Query Language	Not Only SQL
Data Model	Relational Database Management System (RDBMS)	Non-Relational / Distributed Database System
Schema	Suitable for structured data with a predefined schema	Suitable for unstructured or semi-structured data
Data Storage Format	Tables with rows and columns	Collections or documents
Transaction Support	Follows ACID properties (Atomicity, Consistency, Isolation, Durability)	May not fully support ACID; often uses BASE (Basically Available, Soft state, Eventual consistency)
Query Capability	Supports complex queries with JOIN operations	Limited support for JOIN and complex queries
Data Structure	Normalized	Denormalized
Scalability	Primarily vertical scaling (scale-up)	Supports horizontal scaling (scale-out)
Common Use Cases	Financial systems, ERP, legacy systems	Real-time analytics, IoT, and content management
Examples	MySQL, PostgreSQL, Oracle, SQL Server, Microsoft SQL Server	MongoDB, Cassandra, Couchbase, Amazon DynamoDB, Redis

SQL-based serverless databases offer structured data handling with strong consistency and complex querying, ideal for enterprise systems. NoSQL counterparts provide flexible schemas and high scalability, making them suitable for modern, real-time, and large-scale applications.

Key Characteristics of Serverless Databases

Serverless computing introduces a paradigm shift in infrastructure management from application developers[18]. The following key characteristics define serverless platforms:

Functionality and No Operations (NoOps)

Serverless platforms allow developers to write function-level code using familiar programming languages such as Python, JavaScript, and Java. These platforms often include user-friendly IDEs, reducing the complexity of environment setup. Developers can deploy applications simply by uploading code, with no need for manual server or environment configuration.

Auto-Scaling Capabilities

Serverless platforms automatically handle horizontal and vertical scaling based on workload fluctuations. Horizontal scaling involves spinning up new instances (scale-in) or removing them (scale-out), while vertical scaling adjusts allocated resources (scale-up/down) per function. Once a function completes execution, its instance may remain temporarily cached for reuse. If unused, the platform scales it down to zero, saving resources.

Utilization-Based Billing

Serverless computing is billed on a usage basis rather than by reservation. Compatible providers such as AWS Lambda and Azure Functions are billed on both the base of memory and execution time. Functions only incur expenses when activated since these are event-driven. It removes the expense of unused resources and offers a less expensive alternative to the conventional cloud models, which require constant resource retention.

Separation of Computation and Storage

Serverless systems have a decoupled model between computations and storage. Computation occurs in stateless functions, and the data is utilized based on a cloud storage service such as AmazonS3 or Azure Blob Storage [19]. This decoupling allows independent scaling and pricing which can support better performance of bursty or unpredictable workloads through flexible scaling of resources.

Platform-Imposed Limitations

To preserve the efficiency and scalability of serverless platforms, cloud providers impose certain limitations on serverless functions. These include constraints on maximum execution time, deployment package size, memory allocation, and temporary disk space. These constraints differ between providers and are tuned in such a way as to ensure the stability of the platforms and predictable performance at different workloads.

Myths, Misconceptions, and Challenges of Serverless Databases.

The myths and misconceptions can become an impediment to the rate of adoption of enterprise solutions, leading to misunderstandings about scalability, performance, and security. The cold start, lack of control and vendor lock-in challenges affect design choices and compel organizations to scrutinize their use case before adoption.

Myths and misconceptions

Although its adoption is increasing, serverless databases and the prevailing circumstances sometimes receive misconceptions that block their wider application, particularly in the enterprise. Figure 4 shows some myths are below:

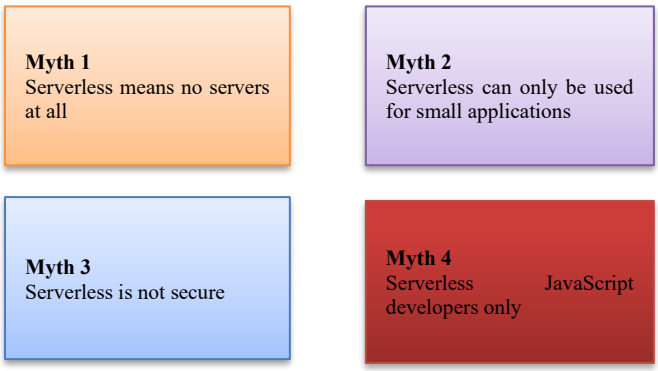

Fig. 4 Myths and misconceptions of the Serverless Architecture

This section addresses common myths and clarifies the realities based on current research and real-world implementations are as follows:

Serverless means no servers at all

The term "serverless" is misleading. No servers are present; they are just abstracted. The developers will never contact the servers directly, as the cloud provider handles all.

Serverless can only be used for small applications

Although serverless is an ideal solution for simple applications and APIs, developers can also use it to drive large-scale solutions. It employs serverless in AI and machine learning integration, real-time messaging applications, video processing, and serverless AI application development frameworks. International companies utilize serverless solutions to support millions of users worldwide.

Serverless is not secure

Security in serverless environments changes, and this does not necessarily mean a worse outcome[20]. Cloud servers such as AWS or Azure have effective security measures; however, developers must still adhere to best practices such as API encryption, API validation, user input validation, and data encryption.

Serverless JavaScript developers only

Serverless also enjoys popularity among Node.js developers, and one reason is the effectiveness of JavaScript in event-driven systems. Nevertheless, other languages, like Python, Go, Java, and .NET, can also be used in serverless environments. Therefore, for instance, if you are in web design services or software development services, then you should not be hampered by the language.

Serverless is Always Cheaper

While serverless platforms offer a pay-per-use pricing model that can lead to cost savings, assuming that serverless is always cheaper is misleading. For workloads with constant or high utilization, traditional or provisioned models may offer better cost efficiency. Furthermore, hidden costs such as cold starts, high I/O charges, and function timeout thresholds must be considered in total cost of ownership calculations.

Cold Start is Always a Major Issue

Cold start latency, where a function experiences delay upon initial invocation, is often cited as a critical drawback of serverless systems. Although cold starts can affect performance, their impact has been significantly reduced through advancements such as pre-warming, provisioned concurrency, and optimized runtime environments. For many applications, especially asynchronous or background tasks, the effect is negligible.

Serverless databases still use servers, but these are abstracted by providers. These are appropriate for both small and large-scale programs, support numerous languages, and provide robust security. It may not be appropriate for high-volume workloads, despite their low cost. The occurrence of cold beginnings is significantly reduced by modern methods.

Key Challenges and Limitations of Serverless Databases

Serverless architectures, while offering significant advantages according to cost-efficiency and scalability, introduce unique security challenges[21]. A thorough evaluation of security best practices is required under the shared responsibility paradigm, in which the underlying infrastructure is managed by the cloud provider. Key risks include:

Shared Infrastructure Vulnerabilities

The multi-tenancy nature of serverless environments increases the potential for data breaches through unintended data leakage or unauthorized access if appropriate security controls are not implemented. Mitigation strategies include leveraging virtual private clouds (VPCs), implementing robust Identity and Access Management (IAM) policies, and utilizing managed identities.

Injection Attacks

Serverless functions are vulnerable to injection attacks (e.g., SQL injection, command injection)[22]. To reduce these dangers, safe coding techniques, frequent security testing (including static and dynamic analysis), and strong input validation are essential.

Data Encryption

The ever-changing nature of serverless settings makes it all the more important to plan and configure thoroughly in order to provide strong encryption for data when it is idle and in transit. The use of server-side encryption mechanisms and appropriate key management strategies is essential.

Performance Degradation

"Cold starts," the initial latency experienced when invoking infrequently used serverless functions, can impact application performance. Mitigation techniques include strategies like function warming (periodic invocation), optimized code, and the use of provisioned concurrency features. Other performance bottlenecks, such as network latency and integration issues, must also be addressed through careful architectural design and performance tuning.

Vendor Lock-In

The reliance on a specific cloud provider's services can create difficulties when migrating to another platform or integrating custom features not supported by the provider.

Cold Start Latency

Cold starts, which take place when a function is called after a period of inactivity, may cause delay for serverless services. This delay can impact user experience, particularly in latency-sensitive applications.

Execution Time Limits

Many serverless platforms impose limits on the execution duration of functions, which may not fit use cases requiring long-running processes.

Benefits and Adoption Landscape for Large Enterprises

Serverless computing transforms large enterprises by simplifying IT operations and supporting agile development. It enables automatic scaling, pay-per-use pricing, and faster deployment, reducing costs and time-to-market. Enterprises leverage serverless for microservices, real-time processing, and AI/ML integration. Adoption is rising across sectors like finance, healthcare, and retail, with many organizations embracing hybrid models to modernize legacy systems and enhance resilience.

Benefits and opportunities of Serverless Computing

Serverless computing offers numerous benefits to its users[23]. These advantages are summed up as follows in this section:

Reduced Operational Overhead

Serverless computing significantly reduces operational complexity. By decoupling infrastructure management from application development, organizations can allocate their resources more efficiently. The developers do not need to worry about the provisioning, scaling, and maintenance of the servers, which makes the processes smoother and quickens the development processes.

Cost Efficiency

Organizations only pay for the resources used during execution using serverless computing's pay-as-you-go pricing model. Serverless computing allows for efficient scalability in response to real-time demand, which in turn reduces operating costs, as opposed to conventional models where organizations pay for supplied instances regardless of use. The financial model has a special advantage to startups and small firms where startup cost can be a major entry limiting factor.

Scalability

Serverless architectures respond to the demand, automatically scaling applications as required; they react to the traffic without any manual scaling. This scalability is essential when dealing with applications with fluctuating load, like e-commerce sites at holiday shopping times [10]. The platform can also automatically allocate additional resources when there is an increase of demand and deallocate resources when there is a decrease in usage.

Faster Time-to-Market

Serverless computing makes the development process faster, as you can deploy quickly. Deployment processes can be simplified, and version control allows quick iteration on features and bug fixes, thereby minimizing the time to market of new apps and functionality. The ability to quickly respond to user feedback and market trends allows organizations to maintain a competitive edge.

Server-side management

The server-side and its administration are not concerns for developers in serverless computing. The management and maintenance of the necessary hardware and software for application deployment is handled by serverless cloud providers [24]. Furthermore, it take care of all administrative tasks so that developers may concentrate on resources like storage, memory, and the CPU.

Easy to deploy

It is simple to deploy serverless apps. For instance, developers just have to upload a few functionalities and produce a new product in order to launch an application. Issues with infrastructure, including server provisioning and scalability, as well as deployment management, will be handled by the serverless.

Adoption Landscape: Industry Trends and Case Studies

Cloud technology developments, changing application needs, and the pursuit of more efficient, scalable, and cost-effective solutions will determine serverless computing's trajectory in the future [25]. The following is a comprehensive analysis of the new developments shaping the future of serverless computing:

Adoption roadmap of serverless database architecture

The successful adoption of serverless database architectures in large enterprises necessitates a structured and strategic approach. This section outlines a phased roadmap that enterprises can follow to adopt serverless databases effectively, are includes:

- It begins with readiness assessment, including workload profiling, infrastructure audits, and compliance checks.
- A pilot implementation is conducted using low-risk use cases like microservices or internal tools to evaluate benefits and limitations.
- This is followed by gradual migration and integration, where serverless solutions are scaled across systems through refactoring and hybrid deployments.
- Finally, monitoring, optimization, and governance ensure performance, cost-efficiency, and compliance. Throughout, best practices such as Infrastructure as Code, CI/CD, observability, and security tools support a smooth and effective transition.

Adoption by cloud Providers

Major cloud providers such as AWS, Microsoft Azure, and Google Cloud have introduced commercial serverless platforms like AWS Lambda, Azure Functions, and Google Cloud Functions[26]. Despite the platforms' scalability and user-friendliness, developers risk being locked into one vendor due to the abstraction of underlying infrastructure elements. In addition, these providers have developed mature serverless database solutions designed to support the scalability, performance, and flexibility needs of modern enterprise applications.

AWS Aurora Serverless

Amazon Aurora Serverless is a configuration tier of the Aurora relational database, available for MySQL- and PostgreSQL-compatible editions. It inherits most features from the standard Aurora engine but offers automatic scaling and cost control. The serverless tier simplifies operations by managing tasks like provisioning, backup, recovery, and failure detection[27]. While it runs as a database cluster within a single region, backups are accessible across regions.

Azure Cosmos DB

Globally distributed and elastic, Azure Cosmos DB is Microsoft's cloud native database solution that can manage JSON documents at Internet scale [28]. It is the primary database service in the Microsoft cloud, with tens of millions of database partitions, over 100 PBs of data under management. Figure 5 shows a serverless database architecture where client requests go through a federation to a load balancer, which routes them to replicas. Each replica runs a database engine with resource control, query processing, and a high-performance Bw-tree++ storage engine.

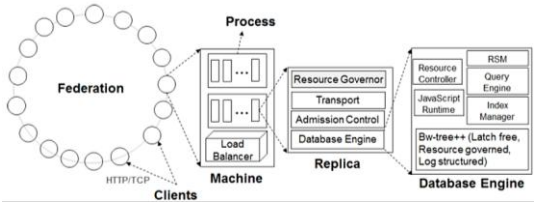

Fig.5 Azure Cosmos DB architecture diagram

Google Cloud Firestore

Google Cloud Firestore is a fully managed, serverless, NoSQL document database offered by Google Cloud. It features a schema-less design, enabling flexible data modeling for dynamic and evolving applications[29]. Firestore can be deployed as either a regional or multi-regional service, with the latter offering enhanced availability and resilience through automatic data replication across multiple geographic locations. It supports real-time synchronization, strong consistency, and seamless integration with other Google Cloud services, making it suitable for high-performance, globally distributed applications.

Real-world use cases of serverless computing in the Enterprise

Studies on serverless adoption in enterprises reveal a growing trend toward integrating serverless architectures into existing IT ecosystems. Industry reports highlight the role of serverless computing in simplifying application development processes, particularly for microservices and event-driven workloads[30]. The discussion below highlights notable use cases, organized by industry sector:

Finance

Serverless computing enables real-time transaction processing and fraud detection in the financial sector. Banks use it to reduce latency and costs, especially during peak periods. It also supports regulatory compliance by streamlining data aggregation and report generation, enhancing speed and reducing infrastructure overhead.

Retail

In the retail industry, Serverless computing enables personalized experiences and efficient supply chain management. It is used in e-commerce web sites on product recommendations, search and check out with a strong track record of scaling to support events of high traffic. Serverless workflows are also combined with IoT in retail where it serves to track inventory in real-time and facilitate efficient ordering, cutting costs and increasing efficiencies.

Healthcare

The serverless computing healthcare sector allows the secure management of patient data, telemedicine, and medical research. It helps with the scalable and fast management of electronic health records and underpins real-time video consultations. Healthcare vendors will enjoy higher reliability and minimal downtime as researchers use big data analysis and simulations with serverless platforms.

Technology

In the technology sector, serverless computing is widely used for building and deploying microservices-based applications. Serverless architectures help tech companies speed up development and improve the reliability of their applications. A primarily software-as-a-service company embraced serverless computing to support user authentication and authorization, and got immediate performance and security. Serverless computing, through the use of streaming data such as IoT or social media, allows real-time analytics and machine learning, among other things. It helps enterprises get insights and make data-driven decisions in a timely manner, facilitating innovation in other areas.

Literature of Review

This section reviews prior research on Serverless Database Architectures: Myths, Benefits, and Real-World Adoption Roadmap for Large Enterprises. Table I presents a structured comparison of these studies, emphasizing their focus areas, methodologies, key findings, objectives, and proposed future directions.

Kondapalli, (2025) explains how serverless databases affect contemporary cloud computing systems by looking at their architectural benefits, implementation issues, and practical uses. An in-depth analysis of the corporate installations proves that there are significant advantages to resource utilization, system availability, and administrative efficiency, in particular, when working with AI and ML workloads. Serverless database technologies are revolutionizing cloud data management by bringing novel ways to infrastructure management and resource optimization. Their systems bring in flexibility never seen before with auto-scaling and consumption prices, and match database prices with use patterns [31].

Mullapudi, (2025) addresses the key aspects, including storage-compute disaggregation, stateless query execution, and distributed transactions management with tackling critical challenges, including cold start latencies and predictability of cost. The article implemented with the detailed analysis of real-life use and study results, shows how serverless databases have changed data management by expanding scalability, better resource utilization and optimizing costs. The article also points out emerging trends in hybrid processing model, advanced caching mechanism and machine learning-based optimization schemes that are defining the future of serverless databases[5].

Zangana, Sallow and Omar, (2024) covers various cloud architectures that overlay functionalities of distributed serverless computing, with some focus on designs like FaaS and concepts of event-driven computing. This paper will explore the principles and benefits of serverless computing by examining how it has transformed the development workflow and infrastructure maintenance. The analysis considers some of the most popular serverless systems and

frameworks and names some of the most important features and aspects like orchestration, scalability, and security. The purpose of this article is to describe how the event-driven and FaaS paradigms are changing cloud computing and allowing developers to create robust and effective apps without the burden of server maintenance. The concept is based on the notion that serverless computing's event-driven designs provide major benefits in terms of scalability, real-time processing, and resource utilization[32].

Zhang, Zhang and Li, (2023) discusses some of the benefits of serverless computing, highlighting impacts on the operational complexities, cost effectiveness, application creation and scalability. It also consider recent research and case study aspects to give full perspective on the benefits and setbacks of implementing serverless architectures. By filling this gap, we expect to give a sketch of the transformative impact of serverless computing on the contemporary business environment and orient future research. Developers can now concentrate on writing code instead of worrying about maintaining infrastructure due to serverless computing, which has become a major paradigm change in the cloud computing space[21].

Wen et al., (2022) provide a thorough assessment of the existing literature outlining the present status of serverless computing research. This publication summarizes 164 papers covering 17 areas of serverless computing research, such as testing and debugging, application migration, performance optimization, programming frameworks, and multi-cloud development. It also provides some interesting research opportunities, trends, and mainstream serverless computing systems. An increasing number of software applications are being developed using serverless computing, a new paradigm in cloud computing. It helps developers to focus on application logic at the level of single functions, sparing them time-wasting and susceptible infrastructure management. Meanwhile, its unique feature offers significant obstacles for the development and implementation of serverless-based applications[18].

Kodakandla, (2021) compares serverless computing comparatively across three crucial aspects cost, scale and performance. The study identifies critical differences in operation characteristics, costing distributions, and real-world effectiveness of the products of leading cloud providers by comparing AWS Lambda, Azure Functions, and Google Cloud Functions. A thorough comparison of performance indicators, such as latency, cold start behavior, and concurrency management then shows the appropriateness of serverless systems for a variety of applications. In addition, the price structures are revealed by a cost analysis, which highlights hidden expenses and the economic consequences of serverless adoption for both small-scale and large-scale applications [33].

Table 2 Literature summary OF Serverless Database Architectures: Myths, Benefits, and Enterprise Adoption

Reference	Focus Area	Approches	Key Findings	Objectives	Future Directions
Kondapalli et.al. (2025)	Architectural advantages and real-world applications of serverless databases	Enterprise-level analysis; cloud deployment in AI/ML workloads	Improved resource utilization, availability, and administrative efficiency; effective for AI/ML tasks	Assess the impact of serverless DBs on modern cloud computing	Explore deeper integration with AI/ML workloads and refine infrastructure automation
Mullapudi et.al. (2025)	Core components and challenges of serverless databases	Analytical review of system components; empirical case studies	Benefits: storage-compute disaggregation, stateless queries; Challenges: cold start, cost predictability	Explain technical building blocks and assess performance implications	Develop hybrid models, caching mechanisms, and ML-based query optimization
Zangana, et.al. (2024)	Distributed serverless cloud architectures (FaaS, event-driven)	Critical literature review of current research and platforms	FaaS promotes scalability, event-driven designs enhance real-time processing and resource utilization	Describe how event-driven architectures reshape cloud development	Improve orchestration, security, and developer-centric design for FaaS
Zhang, et.al. (2023)	Business impact and adoption of serverless architectures	Survey with industry case studies and research synthesis	Simplifies development, improves scalability and cost-efficiency; reduced operational complexity	Provide an overview of benefits/challenges for businesses adopting serverless	Identify gaps in adoption strategy, developer tooling, and long-term cost-effectiveness
Wen et al. (2022)	Comprehensive mapping of serverless computing research directions	Literature review of 164 papers across 17 domains	Serverless eases infrastructure burden; identified research in performance, migration, debugging, multi-cloud, etc.	Characterize state of the art in serverless computing research	Address unresolved challenges in testing, performance tuning, and interoperability
Kodakandla et.al. (2021)	Comparative analysis of major cloud serverless platforms	Benchmarking AWS Lambda, Azure Functions, Google Cloud Functions	Varying performance and pricing; latency and cold starts impact suitability; economic implications vary with scale	Compare performance, scalability, and cost among top providers	Develop unified benchmarking frameworks and transparent cost models

Conclusion and Future Work

A breakthrough in cloud computing, serverless database designs provide businesses more flexibility, scalability, and cost savings in data management. This research shows how the technology has evolved beyond the simple FaaS models into complex, event-driven platform that facilitate a new level of operational efficiency and resource utilization. Serverless databases can help organizations better utilize current resources and nearly eliminate operational overhead by abstracting infrastructure,

dynamically scaling all resources and provisioning automatically. Their move towards cloud-native, modern architectures has seen other significant advantages that include, auto-scaling, pay-per-use, and management. Its functionality in many industries, including those in the financial, retail, healthcare, and technology sectors, has been demonstrating its effectiveness in accelerating the growth process, enhancing performance, and streamlining operations. However, issues such as cold start latency, where vendors risk enjoying lock-in, and threats on their security, as well as the diverse cost trend do occur. These limitations are incrementally being closed by continuous development in distributed computing, observability, and tooling that is cloud-native, further indicating the applicability of serverless databases to enterprise-scale use.

The combination of serverless computing and the edge is expected to be the next step in advancement to support real-time processing and ultra-low latency at the network edge. Further adoption will also be assisted by the continued advancement of platform interoperability, security structures, and multi-cloud compatibility. Future research should be geared towards tackling existing shortcomings and defining best practices as well as finding a mix of serverless ideologies with upcoming technology to include AI, analytics, and event-driven models. In the digital era and beyond, serverless database systems may dismantle traditional barriers to innovation and provide a competitive edge via adaptability, dependability, and a focus on the customer experience.

References

[1] V. P., "Optimizing Serverless Computing: Enhancing Performance and Efficiency in Modern Cloud Architecture," *Int. J. Sci. Technol.*, vol. 16, no. 2, pp. 1–10, Apr. 2025, doi: 10.71097/IJSAT.v16.i2.3054.

[2] T. Naumenko and A. Petrenko, "Analysis of problems of storage and processing of data in serverless technologies," *Technol. Audit Prod. Reserv.*, vol. 2, no. 2(58), pp. 20–25, Apr. 2021, doi: 10.15587/2706-5448.2021.230174.

[3] A. Abhishek and P. Khare, "Cloud Security Challenges: Implementing Best Practices for Secure SaaS Application Development," *Int. J. Curr. Eng. Technol.*, vol. 11, no. 06, Nov. 2021, doi: 10.14741/ijcet/v.11.6.11.

[4] M. Hamza, W. Naeem, and M. Waheed, "Serverless Adoption in Practice : A Socio-Technical Investigation of Motivations , Challenges , and Strategies," *SSRN Electron. J.*, pp. 1–16, 2025.

[5] P. K. Mullapudi, "Serverless Database Architectures: A Deep Dive Into Design Principles And Performance Considerations," *Int. J. Inf. Technol. Manag. Inf. Syst.*, vol. 16, no. 2, pp. 25–35, Mar. 2025, doi: 10.34218/IJITMIS_16_02_003.

[6] V. Singh, "Lessons Learned from Large-Scale Oracle Fusion Cloud Data Migrations," *Int. J. Sci. Res.*, vol. 10, no. 10, pp. 1662–1666, 2021.

[7] V. Kjorveziroski, S. Filiposka, and V. Trajkovik, "IoT Serverless Computing at the Edge: A Systematic Mapping Review," *Computers*, vol. 10, no. 10, pp. 1–21, Oct. 2021, doi: 10.3390/computers10100130.

[8] Siva Prasad Nandi, "Serverless databases: Architectural evolution, implementation strategies, and future directions in cloud-native data management," *World J. Adv. Res. Rev.*, vol. 26, no. 2, pp. 567–574, 2025, doi: 10.30574/wjarr.2025.26.2.1591.

[9] S. Garg, "Predictive Analytics and Auto Remediation using Artificial Inteligence and Machine learning in Cloud Computing Operations," *Int. J. Innov. Res. Eng. Multidiscip. Phys. Sci.*, vol. 7, no. 2, 2019.

[10] J. Saji and A. Kumar, "A Review Paper on Serverless Architecture of Web Applications," *Proc. KILBY 100 7th Int. Conf. Comput. Sci. 2023*, vol. 10, no. 10, pp. 1001–1006, 2023, doi: 10.5958/2278-4853.2021.00966.6.

[11] A. Sharma and S. Kabade, "Serverless Cloud Computing for Efficient Retirement Benefit Calculations," *Int. J. Curr. Sci.*, vol. 12, no. 4, 2022.

[12] V. S. Thokala, "Improving Data Security and Privacy in Web Applications : A Study of Serverless Architecture," *Int. Res. J.*, vol. 11, no. 12, pp. 74–82, 2024.

[13] S. S. S. Neeli, "Serverless Databases: A Cost-Effective and Scalable Solution," *Int. J. Innov. Res. Eng. Multidiscip. Phys. Sci.*, vol. 7, no. 6, p. 7, 2019.

14] M. Menghnani, "Modern Full Stack Development Practices for Scalable and Maintainable Cloud-Native Applications," *Int. J. Innov. Sci. Res. Technol.*, vol. 10, no. 2, pp. 1206–1216, 2025, doi: 10.5281/zenodo.14959407.

[15] B. Boddu, "Serverless Databases Are the Future of Database Management," *J. Sci. Eng. Res.*, vol. 6, no. 1, 2019.

[16] D. Kunda and H. Phiri, "Comparative Study of NoSQL and Relational Database," *Zambia ICT J.*, vol. 1, no. 1, pp. 1–4, Dec. 2017, doi: 10.33260/zictjournal.v1i1.8.

[17] N. Jatana, S. Puri, M. Ahuja, I. Kathuria, and D. Gosain, "A Survey and Comparison of Relational and Non-Relational Database," *Int. J. Eng. Res. Technol.*, vol. 1, no. 6, pp. 1–5, 2012.

[18] J. Wen, Z. Chen, X. Jin, and X. Liu, "Rise of the Planet of Serverless Computing: A Systematic Review," *ACM Trans. Softw. Eng. Methodol.*, vol. 32, no. 5, pp. 1–61, Dec. 2022, doi: 10.1145/3579643.

[19] V. S. Thokala and S. Pillai, "Optimising Web Application Development Using Ruby on Rails , Python , and Cloud-Based Architectures," *Int. J. Innov. Sci. Res. Technol.*, vol. 9, no. 12, pp. 630–639, 2024.

[20] S. P. Godavari Modalavalasa, "Exploring Azure Security Center: A Review of Challenges and Opportunities in Cloud Security," *ESP J. Eng. Technol. Adv.*, vol. 2, no. 2, pp. 176–182, 2022, doi: 10.56472/25832646/JETA-V2I2P120.

[21] W. Zhang, Y. Zhang, and J. Li, "Serverless Computing: A Comprehensive Survey," *ACM Comput. Surv.*, vol. 56, no. 2, pp. 1–38, 2023.

[22] Vikas Prajapati, "Role of Identity and Access Management in Zero Trust Architecture for Cloud Security: Challenges and Solutions," *Int. J. Adv. Res. Sci. Commun. Technol.*, vol. 5, no. 3, pp. 6–18, Mar. 2025, doi: 10.48175/IJARSCT-23902.

[23] V. P. Borade, "Serverless Computing: Benefits, Challenges, and Use Cases," *Int. J. Adv. Res. Sci. Commun. Technol. Int. Open-Access, Double-Blind, Peer-Reviewed, Ref. Multidiscip. Online J.*, vol. 4, no. 1, pp. 480–487, 2024, doi: 10.48175/IJARSCT-15070.

[24] S. P. B. and G. Modalavalasa, "Advancements in Cloud Computing for Scalable Web Development: Security Challenges and Performance Optimization," *J. Comput. Technol. Int. J.*, vol. 13, no. 12, pp. 01–07, 2024.

[25] V. Prajapati, "Cloud-Based Database Management : Architecture , Security , challenges and solutions," *J. Glob. Res. Electron. Commun.*, vol. 1, no. 1, 2025.

[26] Vashudhar Sai Thokala, "Scalable Cloud Deployment and Automation for E-Commerce Platforms Using AWS, Heroku, and Ruby on Rails," *Int. J. Adv. Res. Sci. Commun. Technol.*, pp. 349–362, Oct. 2023, doi: 10.48175/IJARSCT-13555A.

[27] T. Zois, "The Evolution of Serverless Services," *Res. Gate*, no. March, 2021, doi: 10.13140/RG.2.2.33924.86407.

[28] N. Upreti *et al.*, "Cost-Effective, Low Latency Vector Search with Azure Cosmos DB," May 2025.

[29] R. Kesavan, D. Gay, D. Thevessen, J. Shah, and C. Mohan, "Firestore: The NoSQL Serverless Database for the Application Developer," in *2023 IEEE 39th International Conference on Data Engineering (ICDE)*, IEEE, Apr. 2023, pp. 3376–3388. doi: 10.1109/ICDE55515.2023.00259.

[30] T. Fred and W. Joseph, "Real-World Use Cases of Serverless Computing in Enterprises," 2019.

[31] S. V. Kondapalli, "Serverless Database Solutions: The Next Evolution in Cloud Data Management," *Eur. J. Comput. Sci. Inf. Technol.*, vol. 13, no. 15, pp. 110–118, Apr. 2025, doi: 10.37745/ejcsit.2013/vol13n15110118.

[32] H. M. Zangana, Z. B. Sallow, and M. Omar, "Cloud Architectures for Distributed Serverless Computing: A Review of Event-Driven and Function-as-a-Service Paradigms," *Int. J. Artif. Intell. Robot.*, vol. 6, no. 2, pp. 57–64, Nov. 2024, doi: 10.25139/ijair.v6i2.8597.

[33] N. Kodakandla, "Serverless Architectures: A Comparative Study of Performance, Scalability, and Cost in Cloud-native Applications," *Iconic Res. Eng. Journals*, vol. 5, no. 2, pp. 136–150, 2021.