Available online www.jsaer.com

***Journal of Scientific and Engineering Research*, 2019, 6(1):277-282**

Research Article

ISSN: 2394-2630
CODEN(USA): JSERBR

Serverless Databases Are the Future of Database Management

Balakrishna Boddu

Sr. Database Administrator
balakrishnasvkbs@gmail.com

Abstract: The traditional way of managing databases involves setting up complex systems and constant maintenance, which is becoming challenging for modern apps that need to grow easily, be flexible, and save money. Serverless databases solve this problem by removing the need for manual management of the underlying infrastructure.
This article discusses the main benefits of serverless databases, such as their ability to automatically adjust resources based on workload, reduce the amount of operational work, and speed up development processes. It also talks about the different types of serverless databases, where they can be used, and the challenges of adopting them. By understanding these benefits and potential issues, organizations can make better decisions about their database management strategies to meet the changing demands of the digital world.

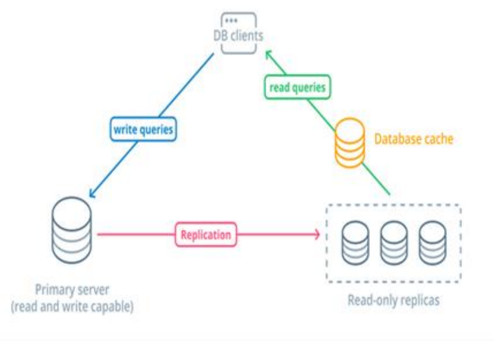
Keywords: Database, AWS, Azure, GCP, atlas, oracle, saas, RDS, aurora, key-value, document, high-availability, scaling.

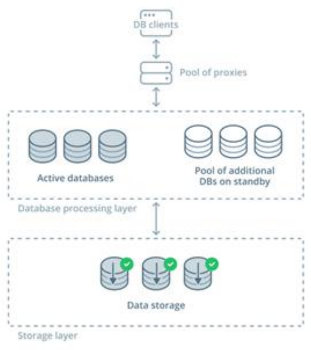
1. Introduction

As technology gets better and companies look for easier ways to handle and increase their data, serverless databases are becoming more and more popular. By getting rid of the need to take care of servers and make them bigger, serverless databases offer a simpler way with less cost. Popular examples of serverless databases include Amazon Web Services DynamoDB, Aurora databases, Cockroach DB, MongoDB Atlas, Microsoft's Azure Cosmos DB, and Google Cloud's Firestore. These Database-as-a-Service (DBaaS) solutions provide businesses with easy-to-use, cloud-based database systems that can be tailored to their specific requirements.
Serverless databases are very flexible and can be used for many different things, like online stores and hospitals. They can handle a lot of data, work well, and are very powerful. Because of this, many businesses that want to manage their data better are choosing to use serverless databases.
The traditional approach to database management, which involves provisioning and managing servers, has become increasingly complex and resource-intensive. Serverless databases, a relatively new paradigm, offer a more scalable, cost-effective, and developer-friendly solution. By abstracting away the underlying infrastructure, serverless databases allow organizations to focus on their core business logic while the platform handles the complexities of scaling and management. This paper explores the key benefits of serverless databases, including their scalability, cost-efficiency, and ease of use. We will also discuss the challenges and considerations associated with adopting serverless databases, such as data consistency, security, and performance optimization.

2. Methodology

Serverless databases are a newer way to use databases that cloud companies offer. They make it even easier for users by taking care of more things. Serverless databases separate the part that stores data from the part that does the searching. This means each part can grow or shrink without affecting the other. In serverless databases, the cloud company takes care of both parts. Traditional databases are the style of databases and database management that has been available for decades. It involves provisioning a physical or virtual server with the appropriate hardware resources and setting it up as a database server. Beyond the initial installation and configuration, this type of setup requires:





Traditional database **vs** **Serverless database**

3. Research background

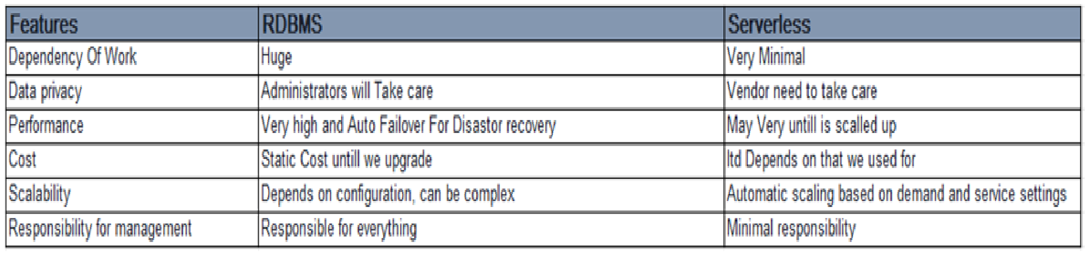
Multiple types of research are going from years to which way we can use serverless databases here are some workarounds. Serverless databases are becoming popular because they can save you money. You only pay for the database when you use it, which can be a lot cheaper than traditional databases. It's also easy to make the database bigger or smaller depending on how much you need it, so you don't waste money. This makes serverless databases great for apps that have a lot of ups and downs in usage. Because they are so flexible, companies can change their apps more quickly to meet new needs or respond to customer changes. This makes serverless databases important for modern app development, as they help companies work more efficiently and come up with new ideas.

Research Advantage:

1. Thomson Reuters is a good example of how serverless databases can be helpful. They used a serverless database to make a tool that looks for suspicious activity in cryptocurrency. This shows how serverless databases can help businesses save money, grow easily, and work more efficiently.
2. Tickets at Ticketmaster reduced its server cost greatly and improved its application performance by migrating its platform to a serverless architecture

Many companies are now using serverless databases, and they will become the standard for database management in the future. The increased flexibility allows organizations to respond more quickly to market changes or adapt to customer needs. These benefits make serverless databases central to modern application development, boosting efficiency and encouraging innovation.

There are some key differences between regular databases vs serverless databases,



Features	RDBMS	Serverless
Dependency Of Work	Huge	Very Minimal
Data privacy	Administrators will Take care	Vendor need to take care
Performance	Very high and Auto Failover For Disastor recovery	May Very untill is scalled up
Cost	Static Cost untill we upgrade	ltd Depends on that we used for
Scalability	Depends on configuration, can be complex	Automatic scaling based on demand and service settings
Responsibility for management	Responsible for everything	Minimal responsibility

As per FMI: The market for serverless apps is already well-developed in North America. In Europe, it's becoming more popular because many retail businesses are using serverless apps. In South Asia and the Pacific, the market for serverless apps is growing quickly due to increasing demand from many industries, like finance, telecommunications, and retail.

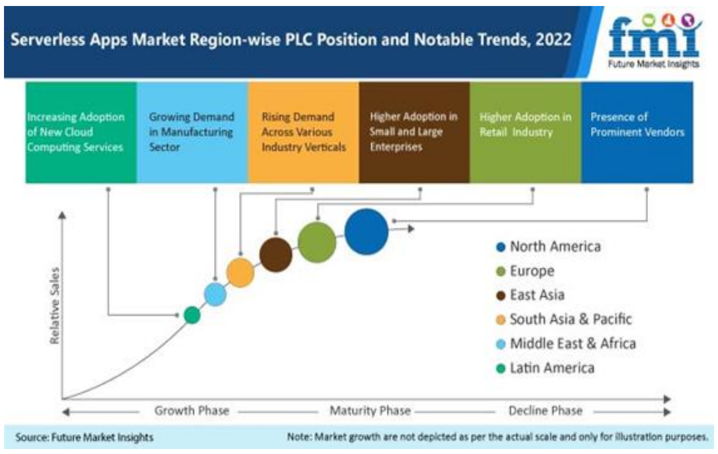

Serverless Apps Market Region-wise PLC Position and Notable Trends, 2022
fmi Future Market Insights
Increasing Adoption of New Cloud Computing Services
Growing Demand in Manufacturing Sector
Rising Demand Across Various Industry Verticals
Higher Adoption in Small and Large Enterprises
Higher Adoption in Retail Industry
Presence of Prominent Vendors
North America
Europe
East Asia
South Asia & Pacific
Middle East & Africa
Latin America
Relative Sales
Growth Phase
Maturity Phase
Decline Phase
Source: Future Market Insights
Note: Market growth are not depicted as per the actual scale and only for illustration purposes.

Diagram: *Serverless App Market as per FMI Report.*

The serverless apps market demand is expected to increase at 23.4% CAGR between 2022 and 2032 in comparison with the 21.3% CAGR registered from 2017 to 2021. With the quick technological shift in the business environment, companies are launching new products and services thus concentrating on reduced time to market to meet the exponentially increasing consumer expectations. The growing importance of such trends is expected to drive the serverless apps market during the predicted period.

There are some kinds of serverless storage layer tools in the Market too. Some of them are very useful for storing raw data like text, CSV, pdf, word, NoSQL data, SQL data, etc. Below are vendors providing serverless storage layers.

Amazon S3: Amazon Simple Storage Service is a storage location.

Azure Data Lake: Microsoft's analytics platform and serverless data lake are offered through the company's public cloud, Azure.

Google Cloud Storage: This RESTful cloud storage solution is offered through the Google Cloud Platform.

Amazon DynamoDB: fully managed NoSQL database service is primarily used for OLTP workloads.

Amazon Aurora: Aurora is a relational database service offered through AWS.

Google BigQuery: BigQuery is commonly used as a serverless data warehouse for BI workloads.

Fauna: Fauna is a geographically distributed transactional database that emphasizes low-latency reads and writes.

Rockset: Designed for real-time analytics at scale, Rockset is a real-time indexing database that powers sub-second search and analytics for modern data applications.

Serverless databases offer a wide range of options to cater to different application requirements. Here are some of the primary types:

1. Document Databases

NoSQL: Stores data in JSON or BSON documents.

Flexible schema: Allows for unstructured or semi-structured data.

Examples: Amazon DynamoDB, MongoDB Atlas, Firebase Firestore

2. Key-Value Stores

Simple data model: Stores data as key-value pairs.

High performance: Optimized for fast reads and writes.

Examples: Amazon DynamoDB, Redis, Memcached

3. Graph Databases

Graph data model: Represents data as nodes and relationships.

Efficient for complex relationships: Ideal for social networks, recommendation systems, and fraud detection.

Examples: Neo4j, Amazon Neptune

4. Time Series Databases

Optimized for time-stamped data: Handles large volumes of time-series data efficiently.

Use cases: IoT, financial data, scientific data.

Examples: TimescaleDB, InfluxDB

5. Relational Databases

SQL-based: Offers a structured data model with tables, rows, and columns.

Hybrid approach: Combines serverless architecture with traditional relational databases.

Examples: Amazon Aurora Serverless, Google Cloud Spanner

Choosing the right type of serverless database depends on your specific application needs, such as:

Data structure: Consider whether your data is structured, semi-structured, or unstructured.

Query patterns: Evaluate the types of queries you'll be performing (e.g., point lookups, range scans, joins).

Performance requirements: Determine the required latency, throughput, and scalability.

Cost considerations: Assess the pricing models and cost implications of different database types.

4. Use Cases for Serverless Databases

Let's dive into some of the use cases for serverless databases

Workloads: Businesses that have apps with a lot of ups and downs in usage can have trouble managing their resources. For example, public events can cause a sudden increase in traffic on social media. Serverless databases can handle these sudden increases in activity without slowing down. They can quickly get bigger or smaller to meet the need, so you don't pay for resources you're not using.

SaaS Database Solutions: Software as a Service (SaaS) providers sometimes manage individual database instances for every customer. While they can place these database instances in a single cluster, they still need to manage each database individually. A serverless database solution allows SaaS vendors to provision database clusters for each customer without worrying about additional costs. When a database is not in use, it will shut down to reduce resource consumption.

CI/CD Implementation: Continuous integration and deployment (CI/CD) are ways to make software development faster and more reliable. Serverless databases work well with CI/CD because they can easily change size to fit the needs of your application. Using serverless databases in your CI/CD process can help you develop and release software more smoothly and with fewer problems.

Auto Scaling: Distributed databases have a clear advantage over traditional databases. Scaling is as simple as adding or removing compute nodes. Remember that we are talking about Serverless environments here; this will be transparent to you, and you won't have to worry about it because it is handled automatically. So, if scalability is important to you make sure you choose a Serverless environment based on a distributed database.

Split Databases: By splitting databases with a serverless database, you can automatically adjust capacity to match demand. A serverless database approach reduces downtime and offers the exact capacity that applications need.

Testing: With serverless databases, you can quickly create a database for temporary projects or tests without having to worry about long-term costs. You don't need to spend time deciding how big the database should be or setting up physical servers. This means you can start using the database right away and learn as you go. You can also easily make the database bigger or smaller as needed. This makes serverless databases a great choice for new businesses that want to test their ideas without spending a lot of money on their database or worrying about how it will grow.

5. Challenges and Security Risk

As businesses get bigger, they need technology that can grow and change easily. Serverless architecture is a popular way to manage databases. It lets businesses use cloud services instead of having their servers.

The biggest advantage of serverless architecture is that it saves money. Businesses only pay for what they use, and they don't have to worry about managing their servers. It also automatically adjusts to handle sudden increases in traffic without problems. Plus, it's easier to maintain because the cloud provider handles things like updates.

However, serverless architecture has some disadvantages. It might be hard to customize the database to fit your exact needs. Also, you might get stuck using a specific cloud provider, which can make it difficult to switch to a different one later.

On the other side of the coin, we need to consider some of the below challenges for serverless databases.

- **Security:** When companies hire another company to manage their servers, they are also trusting that company to keep the servers safe. If the other company doesn't do a good job of protecting the servers, it could cause problems for the company that hired them. Serverless architecture has many different ways that it can be attacked. There are four main security problems that companies often face when they use serverless architecture.
- **Shared Infrastructure:** serverless functions from different applications run on the same infrastructure, which can lead to data leakage and increase the chances of unauthorized access. Companies can also conduct regular security checks to identify potential issues. In addition, most serverless providers now offer solutions for this issue, such as AWS Lambda VPC, AWS Lambda IAM Roles, and Azure Functions Managed Identities.
- **Injection Attacks:** Organizations can adopt secure coding guidelines and provide developers with more in-depth training on how to mitigate these attacks. Organizations can also integrate security testing tools, like static and dynamic analysis, into their serverless architecture and regularly update tools like Guardium to eliminate vulnerabilities.
- **Data Encryptions:** Implementing server-side encryption for databases is challenging due to the nature of serverless architecture.
- **Slow starts:** Organizations can keep functions warm by periodic pinging through tools like AWS CloudWatch Events and Azure Timer Trigger. They can also implement custom strategies to warm up functions before expected traffic spikes. Additional steps include optimizing function code and leveraging resources like provisioned concurrency within AWS Lambda, the Premium Plan within Azure Functions, or minimum instance configuration within Google Cloud Functions.
- **Performance:** A challenge with serverless architecture is latency, which can be caused by several factors, including cold starts, network latency, integration latency, event source latency, infrequently used codes, and memory allocation. Here is how to combat cold starts and event source latency.

6. Cost at Scale

Serverless databases have gained popularity for a good cause. One of the greatest benefits is cost reduction. Unlike traditional databases, serverless databases only charge if something is used, and therefore will save a lot of money. Moreover, scaling up or down is seamless and agile hence much easier to handle the changes in traffic or application needs. This makes serverless databases ideal for applications that experience varying levels of demand. In addition, increased agility allows organizations to be more responsive to changes in the marketplace or swiftly adapt to shifting customer needs. These benefits put serverless databases at the core of modern application development, enhancing efficiency and promoting innovations.

7. Summary

Serverless architecture offers a compelling combination of cost savings, scalability, and agility. Its popularity has surged across various industries, from startups to established enterprises, as implementation has become increasingly streamlined. While challenges related to security, monitoring, and data integrity must be carefully addressed, the potential benefits in terms of time and cost savings, coupled with scalability, make serverless databases a promising choice for many organizations. Before adopting a serverless architecture, thorough research, and consideration of best practices are essential to ensure successful implementation.

References

[1]. **Serverless Architectures on AWS**: With examples using AWS Lambda by Peter Sbarski: This book delves into building serverless applications on AWS, including database integration. It covers design patterns and best practices for serverless architecture

[2]. **Five Book Recommendations for Serverless Developers (article):** This HackerNoon article offers a broader perspective on serverless development, with a section on recommended books. While not specifically focused on databases, these books will equip you with the knowledge to understand how serverless databases fit into the larger serverless architecture picture.

[3]. **Firestore:** The NoSQL Serverless Database for the Application Developer - This paper discusses Google's Firestore, a NoSQL serverless database, and its benefits for application development(https://research.google/pubs/firestore-the-nosql-serverless-database-for-the-application-developer/)

[4]. **Survey on Serverless Computing** - This systematic survey reviews 275 research papers on serverless computing, highlighting its significance in reducing costs, improving scalability, and eliminating server-side management. (https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-021-00253-7)

[5]. **Serverless Cloud Computing:** State of the Art and Challenges - This article explores the serverless model as a paradigm shift in cloud computing, focusing on event-driven execution and usage-based billing. (https://link.springer.com/chapter/10.1007/978-3-031-26633-1_11)

[6]. **Designing Data-Intensive Applications" by Martin Kleppmann:** This book covers the fundamentals of data management in distributed systems, including the latest trends in database architecture such as serverless computing. It’s excellent for understanding how modern databases, including serverless ones, manage scale, reliability, and performance.

[7]. **"Cloud Native Patterns" by Cornelia Davis:** This book provides insights into cloud-native architectures and includes a focus on serverless technologies. It discusses patterns for building scalable and maintainable applications in the cloud, with serverless databases playing a key role.

[8]. **"Database Internals:** A Deep Dive into How Distributed Data Systems Work" by Alex Petrov: While this book provides an in-depth exploration of distributed databases, it also touches on emerging trends like serverless database technologies, making it useful for understanding how serverless databases compare to traditional architectures.

[9]. **"Building Serverless Applications with Google Cloud Run" by Wietse Venema:** This book explores serverless solutions specifically in the Google Cloud ecosystem. It provides examples of building and scaling applications using serverless databases and other serverless technologies.

[10]. **"Cloud Native Java**: Designing Resilient Systems with Spring Boot, Spring Cloud, and Cloud Foundry" by Josh Long and Kenny Bastani: This book is useful if you're working with Java and Spring to develop cloud-native applications. It includes discussions on integrating serverless databases to build scalable and resilient systems.