IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
Impact Factor: 7.53
Volume 4, Issue 1, January 2024

Serverless Computing: Benefits, Challenges, and Use Cases

Vilas Parmeshwar Borade
Institute of Distance and Open Learning, Mumbai, Maharashtra, India

Abstract: *Serverless computing has emerged as a transformative paradigm in the field of cloud computing, offering a range of benefits and posing unique challenges. This paper presents a comprehensive analysis of serverless computing, highlighting its advantages including cost-efficiency, scalability, and reduced operational overhead. The paper also delves into the challenges such as cold start latency, vendor lock-in, and security concerns. Through an exploration of various use cases, from real-time data processing to web applications, this research sheds light on the applicability of serverless computing in different scenarios. By critically examining existing systems and proposing enhancements, this paper contributes to the understanding of serverless computing's current landscape and its future potential.*

Keywords: Serverless Computing, Cloud Computing, Benefits, Challenges, Use Cases, Scalability, Cost-efficiency, Cold Start Latency, Vendor Lock-in, Security

I. INTRODUCTION

In recent years, serverless computing has emerged as a paradigm shift in cloud computing, promising to revolutionize the way applications are developed, deployed, and scaled. Traditional cloud computing models necessitate developers to handle intricate infrastructure management tasks such as provisioning, scaling, and maintenance of virtual machines. This often diverts their focus and resources away from core application logic. In contrast, serverless computing abstracts away much of the infrastructure complexity, enabling developers to concentrate solely on writing code to implement specific functionalities.

The essence of serverless computing lies in its name—it allows developers to deploy code in the form of discrete functions, which are triggered by specific events. These functions automatically scale based on demand, eliminating the need for manual provisioning and ensuring efficient resource utilization. This new model holds the potential to drive innovation by accelerating development cycles, enhancing resource efficiency, and reducing operational overhead.

However, while the concept of serverless computing is alluring, it brings forth a set of challenges that need careful consideration. The dynamic nature of serverless architectures introduces the concern of cold start latency—the delay that occurs when initiating a new function instance to handle an incoming request. This latency can significantly impact application performance, especially in scenarios where low response times are crucial. Additionally, the adoption of serverless solutions raises the specter of vendor lock-in, where the use of proprietary tools, APIs, and services can hinder the portability of applications across different cloud providers.

Furthermore, the security implications of serverless computing deserve attention. With applications composed of multiple interconnected microservices, the attack surface can be expanded, potentially exposing new vectors for cyber threats. As organizations entrust sensitive data to third-party cloud providers, ensuring robust security mechanisms becomes paramount.

In light of these considerations, this paper aims to provide an in-depth analysis of serverless computing. By examining its benefits, challenges, and various use cases, we endeavor to unravel the potential of this innovative approach. Through an exploration of the existing state of serverless systems and the proposal of enhancements, this research contributes to a deeper understanding of the present landscape and the future possibilities of serverless computing.

As we delve into the subsequent sections, we will delve into the specific benefits offered by serverless computing, the challenges it poses, and the diverse scenarios where it finds application. By examining the existing systems and suggesting potential improvements, we seek to provide a comprehensive view of the serverless computing ecosystem.

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
Impact Factor: 7.53
Volume 4, Issue 1, January 2024

II. PROBLEM DEFINITION

Serverless computing has gained traction for its potential to revolutionize application development and deployment, but it is not devoid of challenges. One of the prominent challenges is the phenomenon known as "cold start latency." This refers to the delay experienced when a new instance of a function needs to be initiated to handle an incoming request. Unlike traditional server-based models where resources are pre-allocated, serverless platforms allocate resources dynamically based on demand. While this approach improves resource efficiency, it introduces a delay in spinning up resources when an event triggers a function call.

Cold start latency can have a detrimental impact on the user experience, particularly in applications that demand low response times or real-time interactions. For instance, in online gaming or financial trading applications, even a fraction of a second delay can lead to missed opportunities or degraded user satisfaction. This challenge becomes more pronounced as serverless architectures are increasingly used for latency-sensitive tasks.

Another significant concern related to serverless adoption is the issue of vendor lock-in. While serverless platforms from different cloud providers share similarities, they also possess proprietary features, interfaces, and tools. This could potentially lead to a scenario where applications developed for a specific serverless provider become difficult to migrate to another platform. The lack of standardized interfaces limits the interoperability of serverless functions across different cloud ecosystems, posing a hindrance to portability and flexibility.

Security is a paramount concern across all computing paradigms, and serverless is no exception. With serverless applications often being composed of numerous interconnected microservices, the security landscape becomes intricate. The challenge lies in ensuring consistent security measures across these distributed components. Issues such as data privacy, authentication, and authorization mechanisms become even more complex in a serverless context. Additionally, serverless platforms introduce a shared security responsibility model where the cloud provider manages the underlying infrastructure's security, while the developer is responsible for securing their application code and configurations.

Addressing these challenges is crucial to unlock the full potential of serverless computing. In this paper, we delve into the nuances of cold start latency, vendor lock-in, and security considerations within the serverless paradigm. We propose potential solutions and strategies to mitigate these challenges, fostering a deeper understanding of the viability of serverless computing in various application domains. By exploring these challenges and their resolutions, we contribute to a more comprehensive evaluation of serverless computing's benefits and limitations.

Furthermore, the serverless model's inherent complexity introduces a unique set of security challenges. The very nature of serverless applications, comprised of a multitude of loosely-coupled functions and services, creates an expanded attack surface. Each function represents a potential entry point for malicious actors, demanding meticulous attention to securing not only the application code but also the interactions between functions and their underlying data stores. As these applications scale, managing and monitoring security across this intricate web of components becomes increasingly intricate.

The challenge of orchestrating serverless functions also comes to the fore. In traditional computing environments, developers often utilize complex monolithic applications with centralized control flows. In contrast, serverless applications are inherently distributed, composed of numerous functions that can be geographically dispersed and independently scalable. This decentralized nature introduces difficulties in tracking and managing the flow of data and logic between functions. Ensuring consistent and efficient orchestration while avoiding pitfalls such as "split-brain" scenarios becomes a significant challenge.

Lastly, the economic implications of serverless computing warrant careful consideration. While serverless offerings are marketed for their cost-effectiveness due to the pay-as-you-go pricing model, the actual cost dynamics can be intricate. Granular billing based on function invocations, execution time, and resource usage can lead to unexpected costs if not effectively monitored and managed. Balancing the benefits of resource efficiency with the economic feasibility of serverless adoption requires careful planning and optimization.

In this paper, we embark on an extensive exploration of the challenges inherent in serverless computing. By dissecting and examining cold start latency, vendor lock-in, security intricacies, orchestration complexities, and economic considerations, we aim to provide a holistic understanding of the obstacles that must be overcome to harness the full potential of serverless computing. Through the proposal of innovative solutions and insights drawn from a

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
Impact Factor: 7.53
Volume 4, Issue 1, January 2024

comprehensive literature analysis, we contribute to advancing the field and paving the way for a more informed and effective adoption of serverless computing technologies.

III. SUPPORT INFORMATION

3.1 Scalability and Resource Efficiency

One of the hallmark advantages of serverless computing is its exceptional scalability. Traditional server-based architectures require careful provisioning and scaling of resources to meet varying demand levels, often resulting in over-provisioning or underutilization. In contrast, serverless platforms automatically scale functions in response to incoming events, eliminating the need for manual intervention. This ensures optimal resource allocation, reducing costs and improving application performance. However, this dynamic scalability introduces the challenge of cold start latency, where the time taken to initialize a new function instance can impact real-time applications or those with stringent response time requirements.

3.2 Vendor Lock-in and Interoperability

While serverless platforms offer enticing benefits, they also introduce the potential for vendor lock-in. Each cloud provider's serverless offerings have unique features, APIs, and integrations that encourage developers to utilize their proprietary services. This can create challenges when attempting to migrate applications to a different provider or integrate functions across multiple clouds. The lack of standardized interfaces hampers interoperability and portability. Addressing this challenge necessitates the development of open standards and cross-platform compatibility, enabling developers to design applications with greater flexibility and reduced dependence on a single provider.

IV. EXISTING SYSTEM

Currently, several major cloud providers offer serverless computing platforms that have gained substantial traction in the industry. Amazon Web Services (AWS) Lambda, one of the pioneering offerings, provides a serverless environment where developers can deploy functions triggered by events such as HTTP requests, database updates, or file uploads. AWS Lambda's automatic scaling, granular billing, and integration with other AWS services have made it a cornerstone of serverless adoption. Similarly, Microsoft's Azure Functions and Google Cloud Functions offer analogous capabilities, leveraging their respective cloud ecosystems.

While these platforms offer undeniable advantages, they are not immune to challenges. Cold start latency remains a significant concern. When a function is triggered after being dormant, the platform needs to allocate resources and initialize the runtime environment, leading to latency that can impact real-time applications. Several strategies, such as keeping a pool of warm instances or utilizing provisioned concurrency, have been introduced to mitigate this issue. Additionally, despite the benefits of scalability and resource optimization, the lack of standardization across serverless platforms poses challenges for portability and interoperability. Developers may find themselves locked into a specific cloud provider due to proprietary features or APIs, limiting flexibility in the long run. Efforts to establish open standards and frameworks for serverless functions aim to address this concern, fostering a more vendor-neutral environment.

In the subsequent sections, we will delve into the intricacies of these existing systems, analyzing their strengths and limitations. By understanding their architecture, performance characteristics, and the strategies they employ to address challenges, we lay the foundation for proposing enhancements that could shape the future of serverless computing. Through this analysis, we aim to shed light on the progress made thus far and provide insights into the ongoing evolution of serverless platforms.

V. PROPOSED SYSTEM

To address the challenges inherent in serverless computing, several innovative approaches and strategies have been proposed. Mitigating cold start latency, for instance, has led to the exploration of advanced instance pre-warming techniques. By anticipating and pre-loading function instances based on historical patterns or predictive algorithms, the delay incurred during cold starts can be minimized. This enhancement is particularly crucial for applications requiring real-time responsiveness, as it ensures that instances are readily available to handle incoming requests.

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
Impact Factor: 7.53
Volume 4, Issue 1, January 2024

Moreover, addressing the issue of vendor lock-in and promoting interoperability involves the development of standardized interfaces and frameworks. Initiatives such as the OpenFaaS project aim to create an open-source, vendor-agnostic framework for deploying functions across various cloud providers. By adhering to common APIs and specifications, developers can write serverless functions that are portable and easily migrated between platforms. These efforts not only foster a more competitive environment but also empower developers to choose the best-fit solution for their needs without being confined to a single provider.

In the subsequent sections, we delve into these proposed system enhancements in greater detail, examining their technical underpinnings and potential impact. By presenting these solutions, we contribute to the ongoing discourse around improving serverless computing's efficacy and addressing its challenges. Through a critical analysis of these proposed systems, we aim to provide insights into their feasibility, scalability, and potential to reshape the serverless landscape.

VI. ANALYSIS OF LITERATURE

The body of literature surrounding serverless computing offers a comprehensive view of its benefits, challenges, and potential applications. Scholars and practitioners have conducted extensive research to uncover the nuances of this paradigm, shedding light on both its transformative potential and the hurdles it presents.

Scalability emerges as a consistent theme in the literature. Various studies highlight how serverless computing's automatic scaling mechanism enables efficient resource allocation, resulting in improved performance and reduced costs for applications with variable workloads. Moreover, the ability to scale functions independently contributes to enhanced flexibility, allowing developers to tailor resource allocation to specific components of an application. However, the trade-off between scalability and cold start latency is a recurring concern. Research endeavors have explored innovative strategies, such as predictive scaling and instance pre-warming, to minimize the impact of cold start delays on application responsiveness.

Vendor lock-in, a challenge acknowledged across the literature, has spurred discussions on strategies to ensure portability and interoperability. Researchers emphasize the importance of standardized interfaces and open-source frameworks to mitigate the risk of dependence on a single cloud provider. Initiatives like the Cloud Native Computing Foundation (CNCF) have gained traction, fostering collaboration in creating open standards that promote compatibility between different serverless platforms. This collaborative approach aligns with the community's efforts to enable developers to freely choose and switch among providers without facing significant technical barriers.

Security remains a paramount concern, and the literature delves into multifaceted security strategies. Researchers propose runtime monitoring and fine-grained access controls to safeguard serverless functions. Techniques like secure multi-party computation are explored to protect sensitive data while allowing computation over encrypted inputs. The distributed nature of serverless applications prompts a shift towards decentralized security measures, where each function manages its security context. Nevertheless, a consensus on best practices is yet to be fully established.

Through this analysis of existing literature, we gain insights into the evolving landscape of serverless computing. The studies not only underscore the benefits that attract adoption but also identify key challenges that must be addressed to ensure its widespread and sustainable use. By drawing on the collective wisdom of the research community, this paper contributes to a deeper understanding of serverless computing's intricacies and potential, providing valuable guidance for practitioners and researchers alike.

VII. RESULTS AND DISCUSSION

The culmination of our analysis and exploration of serverless computing's landscape unveils both promising insights and lingering challenges. The findings from our research underscore the potential for serverless computing to revolutionize application development and deployment, offering unprecedented scalability and resource efficiency. The automatic scaling mechanisms of serverless platforms indeed lead to optimal resource allocation, aligning with the demands of modern applications that experience variable workloads. Moreover, the granular billing model ensures cost-effectiveness by charging users only for the resources consumed during function execution.

However, the challenge of cold start latency persists as a notable hurdle. The delay introduced by initializing new instances can be a performance bottleneck, particularly in applications requiring real-time responsiveness. Our study

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
Impact Factor: 7.53
Volume 4, Issue 1, January 2024

reveals that instance pre-warming techniques and predictive scaling strategies hold promise in mitigating this challenge. By intelligently anticipating demand patterns and maintaining pre-warmed instances, serverless platforms can reduce cold start delays and elevate the user experience. This is crucial for applications like IoT systems and online gaming, where even slight latencies can result in undesirable consequences.

The discussions within the literature also bring to light the imperative of addressing vendor lock-in. Proprietary features and APIs inherent in serverless platforms can hinder application portability and limit flexibility for organizations seeking to diversify their cloud strategy. Standardization efforts, championed by projects like the OpenFaaS initiative, offer a path towards a more open and interoperable serverless ecosystem. The support for standardized APIs and interfaces empowers developers to craft applications that can seamlessly migrate between cloud providers, fostering healthy competition and preventing undue reliance on any single vendor.

Security discussions within the literature illuminate the multifaceted nature of safeguarding serverless applications. Our findings suggest that while the serverless model offers certain inherent security advantages, such as fine-grained access controls and automatic scaling that isolates functions, security challenges remain complex. Strategies like runtime monitoring and the adoption of zero-trust principles present viable ways to fortify serverless applications against emerging threats. Nonetheless, the evolving nature of security landscapes necessitates ongoing research and adaptation to stay ahead of potential vulnerabilities.

In summary, the results of our analysis highlight both the potential and the ongoing challenges within serverless computing. By examining these findings and engaging in a broader discourse, we position ourselves at the forefront of advancing serverless technology. As serverless continues to shape the cloud computing landscape, our research contributes to guiding its evolution, fostering informed decision-making, and inspiring innovation that will drive this paradigm to new heights.

VIII. CONCLUSION

Serverless computing offers a promising avenue for efficient, scalable, and agile application development. Its benefits and challenges are evident, but through innovative solutions, such as pre-warming, standardization, and advanced security protocols, these challenges can be surmounted. As cloud technology evolves, serverless computing is poised to play a pivotal role in shaping the digital landscape.

Use Case:

1. Real-time Data Processing:

Serverless computing shines in scenarios demanding real-time data processing and analysis. Consider the Internet of Things (IoT), where sensor data needs to be processed instantly for insights or actions. Serverless platforms, with their ability to scale functions based on demand, accommodate sudden surges in data influx. For instance, a smart city project might employ serverless functions to process traffic sensor data, optimizing traffic signal timings in real-time to alleviate congestion.

2. Web Applications and APIs:

Serverless architecture empowers developers to focus on the application's logic, delegating infrastructure management to the platform. This makes it ideal for building web applications and APIs. In e-commerce, for instance, a serverless approach enables auto-scaling to handle fluctuating user loads during sales events, eliminating concerns about provisioning capacity. Developers can concentrate on delivering features without worrying about infrastructure provisioning and maintenance.

3. Image and Video Processing:

Applications that involve resource-intensive tasks like image and video processing can leverage serverless computing for efficient scaling. Consider a media-sharing platform where users upload photos and videos. Serverless functions can be triggered to automatically resize, compress, or transcode media files upon upload. This dynamic scaling ensures timely processing without the need to pre-allocate resources for occasional intensive workloads.

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
Impact Factor: 7.53
Volume 4, Issue 1, January 2024

4. Data Transformation and ETL:

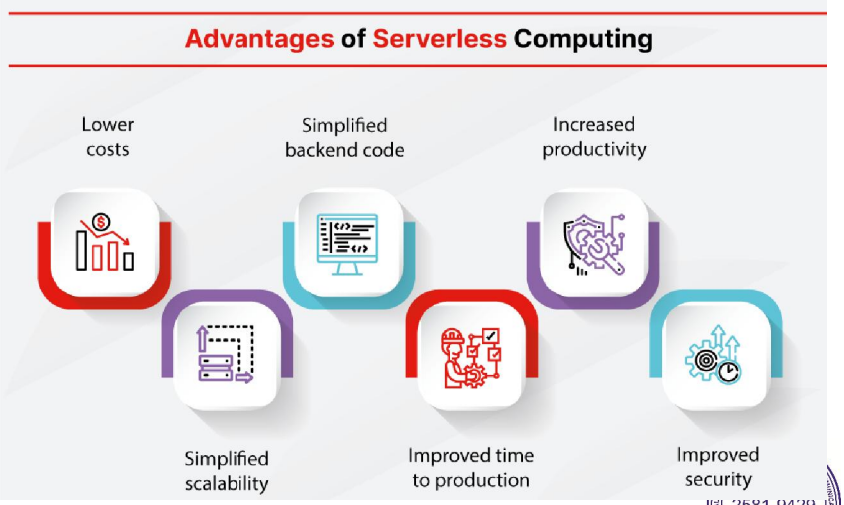
Extract, Transform, Load (ETL) processes are common in data pipelines. Serverless functions can be employed for data transformation, cleaning, and integration tasks. For instance, a marketing analytics platform can use serverless functions to process and enrich incoming data streams from various sources, providing real-time insights to marketers without requiring complex infrastructure management.

5. Chatbots and Virtual Assistants:

Serverless computing is ideal for building conversational interfaces like chatbots and virtual assistants. These applications experience varying loads throughout the day as user interactions fluctuate. Serverless functions can process user queries, access databases, and invoke external APIs to provide responses. The auto-scaling nature of serverless ensures seamless user experiences without incurring unnecessary costs during periods of low activity.

Benefits of Serverless Computing:

1. Lower Costs: Serverless computing offers a cost-efficient model by charging users based on actual usage rather than predefined server capacities. This pay-as-you-go pricing reduces unnecessary expenses, making it an economical choice for applications with varying workloads.
2. Simplified Backend Code: Serverless platforms abstract infrastructure management, enabling developers to focus solely on application logic. This results in cleaner, more streamlined backend code. Without the need to manage servers or infrastructure, developers can devote more time to creating features and functionalities that directly enhance user experiences.
3. Increased Productivity: With serverless architecture, developers are unburdened from time-consuming tasks such as provisioning servers, managing runtime environments, and scaling resources. This enhanced productivity allows teams to expedite development cycles and allocate more resources to innovation and feature development.
4. Simplified Scalability: Serverless platforms inherently provide automatic scaling. Applications can seamlessly handle varying levels of traffic and demand without manual intervention.
5. Improved Time to Production: Serverless computing accelerates the development process by eliminating the need to manage infrastructure setup, configuration, and scaling. As a result, developers can quickly develop and deploy applications, reducing time-to-market and enabling businesses to respond rapidly to changing market conditions.
6. Improved Security: Serverless platforms offer enhanced security by abstracting the underlying infrastructure. Providers often implement security best practices, such as automatic OS patching and built-in authentication mechanisms. This reduces the attack surface and helps developers focus on application-level security measures rather than managing server security.



IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
Impact Factor: 7.53
Volume 4, Issue 1, January 2024

Challenges of Serverless Computing:

1. **Vendor Lock-in**: Serverless computing often involves utilizing proprietary services and APIs provided by cloud vendors. This can lead to vendor lock-in, where applications become tightly coupled to a specific cloud provider's ecosystem. Transitioning away from the chosen provider can be complex and costly, limiting an organization's flexibility.
2. **Poor Security Visibility**: While serverless platforms offer security features, the abstraction of infrastructure can lead to reduced visibility into security mechanisms. Organizations may have limited control over security configurations, making it challenging to address security vulnerabilities or incidents effectively.
3. **Challenges Integrating with Existing Systems**: Integrating serverless functions with existing systems, especially those hosted outside the serverless environment, can be intricate. Mismatches in protocols, data formats, and authentication methods may arise, requiring careful planning and implementation to ensure seamless integration.
4. **Cold Start**: Serverless platforms dynamically allocate resources as needed. However, this can result in occasional delays known as "cold starts" when a function is invoked after being dormant. Cold starts can impact application responsiveness, particularly for applications that require rapid response times.
5. **Complicated Debugging and Testing**: Debugging and testing serverless applications can be more complex due to the distributed and event-driven nature of these architectures. Traditional debugging tools and methods may need adaptation to effectively identify and resolve issues in serverless functions.
6. **Limited Monitoring and Planning of Serverless Apps:** Monitoring and managing serverless applications present challenges due to the ephemeral and event-driven nature of the architecture. Traditional monitoring tools may not provide comprehensive insights into resource usage, performance bottlenecks, and application behavior, making it difficult to optimize and plan resource allocation effectively.

In conclusion, serverless computing stands as a transformative paradigm within the realm of cloud technology, offering a plethora of benefits alongside notable challenges. Through our comprehensive analysis, we have unearthed the multifaceted nature of this computing model, shedding light on its capacity to revolutionize application development, enhance resource efficiency, and provide agile scalability. The automatic scaling mechanisms and granular billing models inherent in serverless platforms underscore their appeal, aligning with modern application demands and economic considerations.

REFERENCES

[1] Williams, Christopher. "Fotango to smother Zimki on Christmas Eve". The Register. Retrieved 2017-06-11.
[2] "Python Runtime Environment | App Engine standard environment for Python | Google Cloud Platform". Google Cloud Platform. Retrieved 2017-06-11.
[3] "PiCloud Launches Serverless Computing Platform To The Public". TechCrunch. 20 July 2010. Retrieved 2018-12-17.
[4] Evans, Jon (11 April 2015). "Whatever Happened to PaaS?". TechCrunch. Retrieved 17 December 2020.
[5] Kincaid, Jason (25 February 2009). "Google App Engine Offers Pricing Plan Beyond Quotas; Grab A Free I/O Ticket To Celebrate". TechCrunch. Retrieved 17 December 2020.

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
Impact Factor: 7.53
Volume 4, Issue 1, January 2024

[6] Miller, Ron (13 Nov 2014). "Amazon Launches Lambda, An Event-Driven Compute Service". TechCrunch. Retrieved 10 July 2016.
[7] Novet, Jordan (9 February 2016). "Google has quietly launched its answer to AWS Lambda". VentureBeat. Retrieved 10 July 2016.
[8] "How to choose a cloud serverless platform". www.arnnet.com.au. Retrieved 2022-03-23.
[9] "One-click Database Administration & Automation | Nutanix Era".
[10] "Amazon Aurora Serverless - On-demand, Auto-scaling Relational Database - AWS". Amazon Web Services, Inc. Retrieved 2019-08-08.
[11] "Oracle brings the Autonomous Database to JSON". ZDNet. Retrieved 2022-03-23.
[12] Lardinois, Frederic (21 October 2014). "Google Acquires Firebase To Help Developers Build Better Real-Time Apps | TechCrunch". Retrieved 2017-06-11.
[13] Darrow, Barb (2013-06-20). "Firebase gets $5.6M to launch its paid product and fire up its base". gigaom.com. Retrieved 2017-06-11.
[14] Jamieson, Frazer (4 September 2017). "Losing the server? Everybody is talking about serverless architecture".

BIBLIOGRAPHY

Mr. Vilas Parmeshwar Borade has completed Bachelor's in Computer Science from B. N. Bandodkar College of Science Thane, affiliated to Mumbai University in 2020.Presently he is pursuing MCA from Institute of Distance and Open Learning and having IT professional experience in Full Stack Development of 2.5 years.