International Journal of Cloud Applications and Computing
Volume 14 • Issue 1 • January-December 2024

Transforming Application Development With Serverless Computing

Suliman Mohamed Fati
https://orcid.org/0000-0002-6969-2338
Prince Sultan University, Saudi Arabia

Mamdouh Alenezi
https://orcid.org/0000-0001-6852-1206
The Saudi Technology and Security Comprehensive Control Company, Saudi Arabia

ABSTRACT

In serverless computing, the developer relinquishes the management of resources to the cloud provider while focusing on improving the application logic and coding. Due to the immense benefits associated with this approach, it has recently attracted the attention of both industry and academy practitioners. Hence, this article explores the transformative impact of serverless computing on the various stages of application development, including design, development, testing, deployment, and maintenance. By examining the specific ways in which serverless computing transforms each stage, the authors uncover the advantages and benefits of using this emerging technology. The research shows that serverless computing accelerates the development lifecycle of applications as compared to traditional server-based architectures. In addition, this study provides valuable insights into the ways in which developers can leverage serverless computing to improve their workflows and productivity.

KEYWORDS

Serverless Computing, Application Development, Development Lifecycle, Acceleration, Productivity

TRANSFORMING APPLICATION DEVELOPMENT WITH SERVERLESS COMPUTING

Serverless computing, or function as a service (FaaS), is a cloud computing model where the responsibility of managing infrastructure is relinquished to the cloud provider while users focus on building applications (Liu et al., 2023; Nastic, 2024; Sewak & Singh, 2018). In the serverless computing environment, applications are broken into smaller functions, with each function performing a specific task (P'erez et al., 2018; Shahrad et al., 2019; Wen et al., 2023). The cloud provider then executes these functions on demand, with no need for a dedicated server. This means that users are charged based on their resource consumption when calling functions without any need for configuring or provisioning an allocated server. The cloud provider automatically scales the number of function instances to handle the workload, ensuring that the application can handle spikes in traffic without impacting performance (Shahrad et al., 2020). Figure 1 shows a serverless computing schematic diagram.

DOI: 10.4018/IJCAC.365288

Figure 1. Serverless computing schematic diagram

The term "serverless" originally referred to peer-to-peer software or client-side solutions, indicating the absence of servers (Ye et al., 2003). However, the background of serverless computing can be traced to the early 2000s when cloud computing was in its infancy (Li et al., 2022). In 2014, Amazon Web Services (AWS) introduced the first event-driven serverless computing platform, AWS Lambda, which allowed developers to write and deploy code to the cloud without worrying about the underlying infrastructure (Chapin & Roberts, 2020; Fylaktopoulos et al., 2016). During this AWS re:Invent event, an international cloud computing conference, the concept of serverless was introduced to form the current serverless landscape. Since then, this model has gained considerable attention as various cloud providers, industrial companies, and academic institutions have developed their own serverless platforms.

Without a doubt, serverless computing represents a natural progression following the advancements and adoption of virtual machines and container technologies (Li et al., 2022; Rajan, 2018). Each step in the abstraction layers has led to more lightweight units of computation in terms of resource consumption, cost, and speed of both development and deployment (Castro et al., 2019). Moreover, serverless builds on long-running trends and advances in distributed systems, publish-subscribe systems, and event-driven programming models, including actor models, reactive programming, and active database systems (Castro et al., 2019; Saba et al., 2023; Shafiei et al., 2022). Recently, the serverless computing model has come with significant traction due to its potential to reduce costs, increase scalability, and improve deployment speed (Chaudhry et al., 2020).

As such, the importance of serverless computing lies in its potential to revolutionize the way applications are developed and deployed (Rajan, 2018). With serverless computing, developers can focus on writing code without worrying about the underlying infrastructure. Traditional application development, however, requires a significant amount of time and resources, including provisioning and managing servers, configuring infrastructure, and deploying code (Shafiei et al., 2022). With serverless computing, developers can write and deploy code in a matter of minutes, without the need for extensive testing and debugging. This is because the cloud provider manages the infrastructure, including scaling and optimization, which allows developers to focus on writing code. Additionally, serverless computing platforms provide a range of features, such as automated deployment, continuous integration and delivery, and monitoring and logging, which can further speed up the development process (Kumari et al., 2021).

This paradigm shift allows for faster development cycles and efficient time-to-market, which can give businesses a competitive advantage by accelerating application development (Wen et al., 2021). However, the question remains: Does it speed up application development?

While serverless computing has the potential to speed up application development, it is not without its challenges. For example, developers must ensure that their code is optimized for serverless

computing, requiring significant changes to their existing codebase. Additionally, serverless computing platforms face limitations like vendor lock-in, making it difficult for developers to move their applications to another cloud provider. Therefore, while serverless computing has the potential to speed up application development, it is important for developers to carefully consider the challenges and limitations of this model before adopting it for their applications.

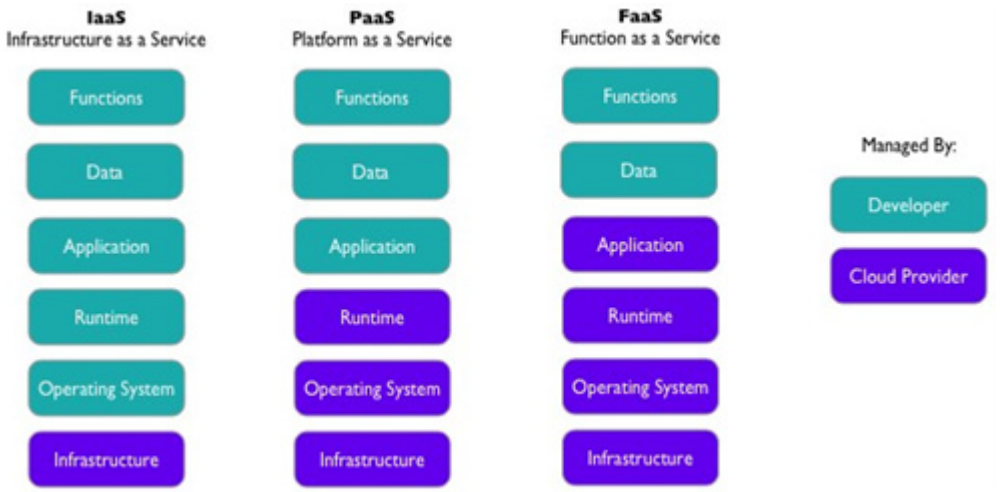
BACKGROUND AND LITERATURE REVIEW

Cloud Computing and the Serverless Model

Cloud computing has gained extensive popularity as a revolutionary approach to providing computing services over the Internet. This computing paradigm encompasses diverse manifestations, such as infrastructure as a service (IaaS), platform as a service (PaaS), software as a service (SaaS), and more—collectively referred to as anything as a service (XaaS). Among these, three prominent models stand out: (1) IaaS; (2) PaaS; and (3) FaaS. Each of these models presents unique ways to leverage cloud resources effectively.

IaaS provides users with virtualized computing resources—such as servers, storage, and networking—which they can configure and manage themselves (Khan et al., 2023). PaaS offers a complete platform for developing, running, and managing applications, including tools, libraries, and infrastructure. In contrast, FaaS embodies the serverless computing architecture, providing a way to run small code snippets, called functions, in response to events without the need to provision or manage servers. In FaaS, the cloud provider dynamically manages the allocation and provisioning of computing resources, eliminating the need for developers to handle server management tasks and enabling them to focus on writing and deploying code. Each model offers different benefits and trade-offs. Therefore, choosing the right one depends on the specific needs of the project or application (Jain et al., 2020). Figure 2 shows a comparison between the three models.

Figure 2. Comparing IaaS, PaaS, and FaaS



In a serverless architecture, applications are broken into smaller, modular functions, allowing the cloud provider to automatically scale the number of function instances to handle the workload. This ensures that the application can manage spikes in traffic without impacting performance. Various

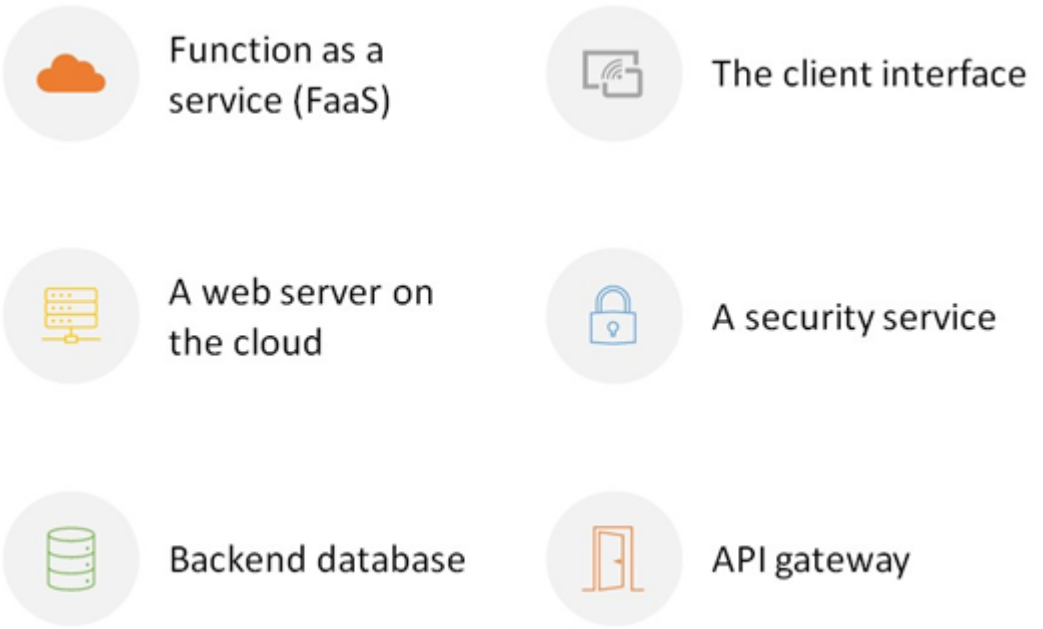
serverless computing platforms are available today, such as AWS Lambda, Azure Functions, IBM Bluemix/OpenWhisk, Alibaba Functions, and Google Cloud Functions (Chowhan, 2018). These platforms offer managed services to developers, enabling them to build applications without worrying about the underlying infrastructure (Achar, 2021). By following this approach, software developers can focus on writing code, while cloud providers take care of provisioning and managing servers, configuring scaling policies, and monitoring infrastructure performance. This paradigm shift allows developers to reduce development time and improve productivity (Mampage et al., 2022), ultimately enabling faster and more agile application delivery.

Serverless computing architecture revolutionizes application development by abstracting away infrastructure management, enabling developers to focus on core application logic (Aditya et al., 2019). Its event-driven execution, on-demand scalability, pay-per-use model, and simplified architecture make it a compelling choice for building modern, efficient, and cost-effective applications (Wang et al., 2020). As serverless technologies evolve and mature, their adoption is expected to expand across a wide range of application domains, transforming how applications are developed, deployed, and operated.

The main components of serverless architecture are depicted in Figure 3, which shows six key components:

- **FaaS:** FaaS is the building block of serverless computing, responsible for executing the logic that determines how resources are allocated within a scenario.
- **Client Interface:** The client interface plays a major role in serverless functionality. It is designed to support bursts of requests, stateless interactions, and flexible integrations.
- **Web Server on the Cloud:** This server initiates stateless interactions after the user triggers them and before the FaaS service terminates them.
- **Security Service:** Security is a key element of serverless operations, typically involving a token service where temporary credentials are generated for users to invoke functions securely.
- **Backend Database:** The backend database stores the information to be shared with the user.
- **Application Programming Interface (API) Gateway:** The API gateway connects the client interface and FaaS, facilitating their communication.

Figure 3. Key components of serverless architecture



Serverless computing is a versatile solution with various use cases. In microservices architectures, functions act as independent services that can be easily scaled and managed individually. Additionally, serverless computing is well-suited for real-time processing, making it an ideal choice for handling time-sensitive data streams like Internet of things data or social media updates, enabling real-time processing and response. It also offers a scalable and cost-effective solution for online and mobile application backends. Moreover, serverless computing efficiently handles data processing, transformation, and analytics tasks, especially for large datasets. Lastly, it simplifies task scheduling and execution by replacing traditional cron jobs, making it a convenient option for managing scheduled tasks.

However, serverless computing also presents challenges, such as limited control over the execution environment, cold start latency, function duration limits, and vendor lock-in. Developers must carefully consider these factors when designing serverless applications.

Transforming Application Development With Serverless Computing

Transforming application development with serverless computing involves leveraging cloud-based execution models to build applications quickly and efficiently. Serverless computing allows developers to focus on application logic by freeing them from tedious infrastructure management tasks (Eismann et al., 2021).

One of the key benefits of using serverless computing in application development is increased scalability (Wen et al., 2021). With serverless computing, applications can seamlessly scale to handle varying levels of traffic and load without manual intervention, ensuring optimal performance even during peak periods. Another advantage is streamlined development processes. The application development process based on serverless architecture is simpler compared to traditional architecture (Wen et al., 2021). Developers only need to write code, build products based on specifications, and deploy them online, which simplifies the development process and allows for faster time-to-market.

Cost-effectiveness is also a significant benefit of serverless computing. By eliminating the need for organizations to manage their own server infrastructure, serverless computing significantly reduces hosting and maintenance costs (Wang et al., 2020). This makes it a cost-effective solution for application development. Additionally, serverless computing improves developer productivity. Developers can focus on writing and deploying their code without the need to manage, maintain, or scale any underlying infrastructure. This allows them to innovate and optimize the functionality and business logic of their applications.

Serverless computing is also language-agnostic (Liu et al., 2023). It supports various programming languages, such as Java, Python, JavaScript, or Node.js. Developers can code in the language or framework they are comfortable with, making it flexible and accessible. Furthermore, serverless computing simplifies deployment and DevOps processes (Woods et al., 2021). Developers do not have to spend time defining infrastructure requirements for integration, testing, delivery, and deployment. This streamlines the development and DevOps cycles, allowing for faster and more efficient application development.

Despite these advantages, developers face challenges in serverless application development. Wen et al. (2021) conducted an empirical study to understand the challenges faced by developers in creating serverless-based applications. The researchers analyzed questions from Stack Overflow, identifying the increasing popularity trend and high difficulty level of serverless computing for developers. They reported various challenges, including design challenges, implementation challenges, deployment challenges, testing and debugging challenges, integration challenges, and security and privacy concerns. The findings highlight the need for better documentation, guidance, and tools to improve the application development experience using serverless computing.

Moreover, rearchitecting existing codebases for serverless computing can be a significant effort and cost (Jin et al., 2021). Transforming applications into optimized serverless orchestrations can help improve application performance and reduce cold start times (Scheuner & Leitner, 2019). Integrating codebase restructuring and optimized serverless orchestration transforms source code into optimized serverless orchestrations based on optimization models such as cost optimization.

Considering the economics of serverless computing is crucial before adopting it for a particular application or workload (Liu & Niu, 2023). Although some cloud providers offer seemingly generous amounts of serverless computing at no charge, the costs can quickly grow for moderate workloads. Therefore, it is important to carefully evaluate the economic implications of serverless computing for each application and workload.

To study serverless computing without vendor or technology-specific assumptions, Gabbrielli and Guidi (2019) proposed a formal programming model for serverless computing. The model combines ideas from the Lambda calculus and process calculi to provide a core formalism for serverless computing. This research aims to abstract the underlying hardware infrastructure and software runtime when building distributed cloud applications.

Efficient graphics processing unit (GPU) sharing for serverless workflows is another important aspect of serverless application development. Satzke et al. (2020) proposed an extension to the open-source KNIX high-performance serverless framework to enable the execution of functions on shared GPU cluster resources. By leveraging shared GPU resources, the extended KNIX framework offers a high-performance and scalable solution for serverless computing applications that require GPU acceleration.

Overall, serverless computing transforms application development by offering increased scalability, streamlined processes, cost-effectiveness, improved developer productivity, and support for various programming languages. Despite the challenges, ongoing improvements in supporting resources and facilities are expected to further enhance the adoption and effectiveness of serverless computing in application development. Researchers and practitioners can explore various directions, including efficient resource sharing, optimization models, and formal programming models, to improve the development experience and facilitate the growth of serverless computing. By addressing these

challenges and leveraging the advantages of serverless computing, developers, researchers, and cloud providers can benefit from this paradigm shift in application development.

RESEARCH METHODOLOGY

This study seeks to address the following research questions:

- **RQ1:** What are the specific ways in which serverless computing transforms the various stages of application development, such as design, development, testing, deployment, and maintenance?
- **RQ2:** To what extent does serverless computing accelerate the development lifecycle of applications compared to traditional server-based architectures?

To answer these questions, a comprehensive research methodology was adopted, comprising two primary components: a literature review and case studies. This dual approach provided both a theoretical foundation and practical insights into the impact of serverless computing on application development.

Literature Review

An extensive literature review was undertaken to consolidate existing knowledge and insights from scholarly articles, technical reports, industry publications, and relevant studies focused on serverless computing and its influence on application development. Publications were carefully selected based on their relevance to the research questions, with a particular emphasis on studies that examined how serverless computing affects the various stages of application development. Both empirical research and theoretical analyses were included to ensure a well-rounded perspective.

Key information was extracted from the selected literature regarding the impact of serverless computing on the design, development, testing, deployment, and maintenance stages of application development. These findings were organized thematically to directly address RQ1. Qualitative analysis techniques were employed to identify common patterns, themes, and insights across the literature, facilitating a comprehensive understanding of the subject matter.

Case Studies

To complement the literature review and obtain practical insights, case studies were conducted involving organizations that have implemented serverless computing in their application development projects. The case studies were selected based on the following criteria:

- **Diversity of Domains:** Organizations from various industry sectors were included to ensure the findings are applicable across different contexts.
- **Variety of Application Types:** Projects involving different types of applications, such as digital applications and data processing tasks, were considered to assess the impact on diverse application architectures.
- **Accessibility:** Organizations willing to participate in the study and share detailed insights about their adoption of serverless computing were selected.

Data collected from these case studies encompassed development timelines, resource utilization, cost analysis, and performance metrics. This information was analyzed to evaluate and quantify the extent to which serverless computing accelerates the development lifecycle of applications compared to traditional server-based architectures, thereby addressing RQ2.

Data Analysis

The data obtained from both the literature review and case studies were subjected to qualitative and quantitative analysis techniques.

- **Qualitative Analysis:** This analysis involved identifying themes, patterns, and key insights from the reviewed literature and case studies. Thematic organization facilitated the exploration of how serverless computing transforms different stages of application development (RQ1).
- **Quantitative Analysis:** Statistical methods were used to compare performance metrics, development timelines, and cost factors between serverless computing and traditional server-based architectures. This analysis provided measurable evidence on the acceleration of the development lifecycle (RQ2).

The analyzed data were synthesized to present, discuss, and interpret the findings in relation to the research questions. The results elucidate the transformative effects of serverless computing on application development stages and quantify its impact on development speed and efficiency. Several limitations of the research methodology are acknowledged:

- **Sample Size:** The number of case studies is limited, potentially affecting the generalizability of the findings.
- **Data Availability:** Access to detailed data relied on the willingness of organizations to share information, which may have resulted in incomplete data.
- **Subjectivity in Qualitative Data:** The analysis of qualitative data may be subject to researcher bias although efforts were made to minimize this through careful coding and validation.
- **Rapid Technological Changes:** Serverless computing is a rapidly evolving field; thus, the findings may not account for the most recent developments.
- **Contextual Factors:** Variations in organizational culture, team expertise, and project management practices could influence the impact of serverless computing. These factors were not controlled for in this study.

Conclusion

Despite these limitations, the combination of a thorough literature review and diverse case studies offers valuable insights into the impact of serverless computing on application development. This methodology enables a robust investigation of the research questions, enhancing the understanding of how serverless computing transforms the development lifecycle and accelerates application development compared to traditional server-based architectures.

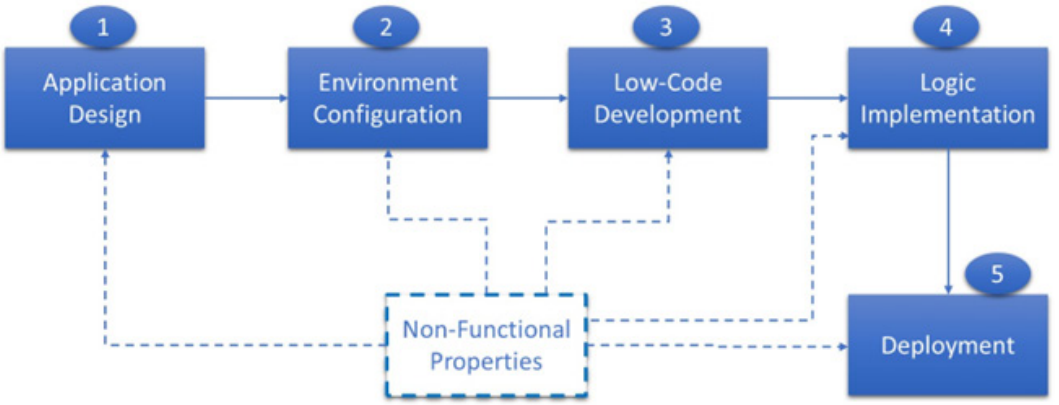
RESULTS

This section illustrates the findings of the current research methodology to answer the two research questions.

Transformative Impact of Serverless Computing on Application Development Lifecycle (RQ1)

Serverless computing, particularly through models like FaaS, has significantly transformed the various stages of application development—design, development, testing, deployment, and maintenance. This transformation has introduced substantial changes in the software development lifecycle, leading to improved efficiency and agility. Figure 4 illustrates the steps in the serverless-based application development process.

Figure 4. Serverless-based applications development process



1. **Shift in Focus from Infrastructure to Code:** One of the most profound impacts of serverless computing is the shift from infrastructure management to coding. Developers no longer need to provision or maintain servers because these responsibilities are managed by the serverless platform (Pu et al., 2019). This paradigm shift reduces the time and effort required for application development, allowing developers to concentrate on writing functional code rather than dealing with infrastructure complexities.
2. **Enhanced Agility and Modularity:** Serverless architecture promotes agility through its support for small, modular functions that can be deployed independently (Woods et al., 2021). This capability enables quick experimentation and iteration, allowing developers to implement changes rapidly without affecting the entire application. The modularity fosters a more iterative development approach, where features can be developed, tested, and deployed in isolation.
3. **Reduction in Operational Overhead:** By offloading operational tasks like server management, scaling, and monitoring to the cloud provider, serverless computing significantly reduces the operational overhead for developers (Wen et al., 2021). This allows teams to focus on delivering new features rather than maintaining infrastructure (Hassan et al., 2021). Consequently, the overall efficiency of the development process improves.
4. **Adoption of Event-Driven Architecture:** Serverless computing introduces event-driven architecture as a core aspect of application design (L'opez et al., 2020). Functions are triggered by specific events, such as user requests or database changes, requiring developers to design applications around event handling. This shift promotes a more reactive design approach and enables applications to scale dynamically in response to varying workloads.
5. **Emphasis on Statelessness:** In serverless architectures, functions are typically stateless, meaning they do not retain data between executions (Rajan, 2018). This necessitates the use of external storage solutions for state management, leading to a more modular and scalable architecture. The focus on statelessness encourages best practices in application design, enhancing reliability and scalability.
6. **Streamlined Integration of Third-Party Services:** Serverless architecture facilitates seamless integration with third-party services through APIs (Wen et al., 2023). Developers can easily invoke functions from other services, promoting a more modular application design. This extensibility allows for the rapid incorporation of new features and services without extensive refactoring.
7. **Continuous Integration and Deployment (CI/CD):** The implementation of CI/CD pipelines is greatly simplified in serverless environments (Wen et al., 2023). Serverless functions can be

deployed automatically, minimizing the risk of disruptions during updates. This leads to faster time-to-market and greater reliability in software delivery.

8. **Shift in Security Paradigms:** Security practices also evolve within serverless architectures. The focus shifts from network-level security to securing individual functions and managing data encryption and authentication (Chawla, 2021). Developers must adopt new security practices tailored to the serverless environment to mitigate risks associated with function execution.
9. **Need for Specialized Monitoring and Debugging:** The transition to serverless computing necessitates the development of specialized monitoring and debugging tools. Traditional monitoring solutions may not effectively support serverless applications, requiring tools like AWS X-Ray or Google Cloud Debugging for performance tracking and troubleshooting.

In summary, serverless computing has reshaped the application development lifecycle by:

- Reducing the burden of infrastructure management
- Enhancing agility through modular, event-driven design
- Streamlining operational tasks and integration with external services
- Facilitating CI/CD practices and evolving security measures
- Introducing the need for specialized monitoring tools

These transformations contribute to faster development cycles, increased scalability, and improved overall efficiency in application development. Table 1 summarizes the differences between traditional and serverless architectures across key development phases.

By examining the transformations brought by serverless computing across these stages, we gain insights into its potential to revolutionize application development practices, enhancing both developer productivity and application performance.

Table 1. Comparison of traditional and serverless architecture in development phases

Application Development	Traditional Approach	Serverless Architecture
Infrastructure management	Developers must provision, configure, and maintain servers	Cloud provider manages infrastructure, freeing developers to focus on code
Scalability	Manual scaling required to handle fluctuating workloads	Automatic scaling based on demand, ensuring optimal resource utilization
Deployment	Complex deployment processes involving manual configuration and Testing	Streamlined deployment process with minimal configuration and automated testing
Maintenance	Regular maintenance tasks required to keep servers up-to-date and secure	Cloud provider handles maintenance and security updates, reducing operational overhead
Cost	Pay-as-you-go model for server resources, leading to unpredictable costs	Pay-per-use pricing for code execution, resulting in cost-effective usage
Developer focus	Developers spend time managing infrastructure, reducing time for application development	Developers can focus on writing code, improving productivity and innovation

continued on following page

Table 1. Continued

Application Development	Traditional Approach	Serverless Architecture
Time-to-market	Lengthy development cycles due to infrastructure management and deployment complexities	Faster development cycles and quicker time-to-market due to simplified architecture
Overall agility	Limits agility due to manual infrastructure management and maintenance	Promotes agility by enabling rapid development, deployment, and scaling

Serverless Computing Accelerates the Development Lifecycle (RQ2)

Serverless computing has been shown to significantly accelerate the application development lifecycle compared to traditional server-based architectures. Empirical evidence highlights several key benefits of serverless architecture, including reduced infrastructure management, automatic scaling, pay-per-use pricing, and event-driven programming. These factors collectively contribute to enhanced development speed and productivity.

A comprehensive literature review on serverless computing (Wen et al., 2022) identifies numerous advantages, such as decreased time for deployment and increased developer focus. However, it also recognizes challenges like cold starts, debugging complexities, vendor lock-in, and security issues. Addressing these challenges is critical for maximizing the potential benefits of serverless architectures. It is an area ripe for future research.

Further analysis, including a study on developers' experiences (Wen et al., 2021), categorized the challenges faced in serverless development. Key issues identified include performance optimization, framework selection, application migration strategies, multi-cloud support, cost management, and effective testing methodologies. Understanding these challenges is essential for optimizing serverless application engineering.

Surveys and studies have underscored the tangible benefits of serverless architectures. For example, a survey by the Cloud Native Computing Foundation (*Graduated and incubating projects,* n.d.) revealed that serverless architectures can reduce deployment times by as much as 70% compared to traditional setups. Additionally, research conducted by AWS (Mezzalira et al., 2022) indicated that developers could cut maintenance and operational time by up to 50%, allowing them to concentrate on coding and feature delivery.

Quantitative analyses demonstrate that serverless architectures can enable developers to build applications two to five times faster than traditional models (*Serverless vs. traditional architecture: What are the differences?,* 2019). This acceleration is primarily due to the elimination of server provisioning and management burdens, allowing developers to focus entirely on application logic. Moreover, serverless computing fosters scalability, cost-efficiency, and quicker release cycles (*Why use serverless computing? Pros and cons of serverless,* n.d.).

However, it is crucial to acknowledge that serverless computing is not a one-size-fits-all solution. Applications requiring long-running processes or intensive computations may be better suited to traditional infrastructures (Ellingwood, n.d.).

Several companies have reaped substantial benefits from adopting serverless architectures. Organizations like Thomson Reuters, iRobot, FINRA, Autodesk, and Square Enix have reported improvements in development speed, cost efficiency, scalability, and overall reliability due to their shift to serverless solutions (*AWS Lambda – Case Studies,* 2022).

In conclusion, empirical evidence supports the assertion that serverless computing accelerates the application development lifecycle compared to traditional architectures. Despite the inherent challenges, serverless architectures have been shown to enhance development speed, increase productivity, improve performance and reliability, reduce costs, and elevate developer satisfaction.

However, the impact of serverless on development speed and quality must be evaluated with consideration for specific factors, including application type, complexity, platform choice, and the skill level of the development team.

Netflix's transition to a serverless architecture via AWS Lambda in 2017 resulted in a remarkable 70% reduction in development time, shifting timelines from months to weeks (*Netflix & AWS Lambda case study*, 2014; Retter, 2020). By outsourcing infrastructure management to AWS, Netflix's engineers could focus on innovation, leading to quicker deployment of code changes—from hours or days to mere minutes.

This transition not only enhanced scalability during peak traffic but also resulted in significant cost savings through a pay-per-use model.

Coca-Cola adopted serverless architectures to develop a new mobile application. The development was completed in just six weeks, a significant improvement over traditional timelines (Rehemagi, 2020). With AWS Lambda managing the infrastructure, developers concentrated on coding, leading to a successful launch that garnered millions of downloads. The efficiency gained through serverless architectures has prompted Coca-Cola to continue leveraging this model for subsequent applications.

Neiman Marcus aimed to develop "Connect," an omnichannel digital selling application, leveraging serverless architecture on AWS. This approach accelerated their launch by over 50% compared to their initial four-month plan (*Optimizing enterprise economics with serverless architectures*, 2021). The serverless model reduced development costs while enabling rapid updates and elastic scalability, ultimately enhancing customer service and associate productivity.

Through these cases, it is evident that serverless computing not only accelerates development cycles but also enhances overall application performance and cost efficiency, paving the way for organizations to focus on their core business objectives.

One of the significant challenges associated with serverless computing is the risk of vendor lock-in. Serverless platforms like AWS Lambda, Azure Functions, and Google Cloud Functions offer a range of proprietary services and APIs that, while enhancing functionality, can make it difficult to migrate applications between providers.

For example, AWS Lambda integrates tightly with other AWS services, such as Amazon S3, DynamoDB, and API Gateway. These integrations often use AWS-specific configurations and triggers, making the application's architecture dependent on the ecosystem of AWS. Similarly, Azure Functions may leverage Azure-specific services like Azure Blob Storage or Azure Event Hubs, creating a reliance on Azure's platform-specific features.

This tight coupling can pose challenges when attempting to migrate applications to a different cloud provider. Rewriting code, reconfiguring services, and adapting to different APIs can be time-consuming and costly. For instance, a function utilizing the AWS DynamoDB would need significant adjustments to work with Azure's Cosmos DB or Google's Cloud Firestore.

CONCLUSION

This study has thoroughly examined the transformative impact of serverless computing on application development, spanning design, development, testing, deployment, and maintenance phases. By integrating a comprehensive literature review with three detailed case studies from diverse industry sectors, the authors have elucidated how serverless architectures fundamentally alter development processes, leading to accelerated lifecycles and enhanced operational efficiencies.

This analysis reveals that serverless computing significantly benefits developers by allowing them to concentrate on writing code without the overhead of managing infrastructure. This shift not only streamlines workflows but also fosters greater innovation and agility within development teams. For managers, serverless computing offers cost reductions, improved scalability, and optimized resource utilization, enabling more strategic allocation of organizational resources.

The case studies provided concrete examples of successful serverless adoption, showcasing tangible improvements in development speed, cost-effectiveness, and performance. These real-world applications underscore the practical advantages and address potential challenges associated with transitioning to serverless architectures. Furthermore, the alignment between literature findings and case study results reinforces the reliability of our conclusions.

Despite acknowledging limitations like a limited number of case studies, potential data availability constraints, and the rapidly evolving nature of serverless technologies, this research offers robust insights into the benefits and transformative potential of serverless computing. The methodology employed, combining qualitative and quantitative analyses, ensures a comprehensive understanding of the subject matter.

For practitioners, researchers, and organizations involved in application development, the findings advocate for the adoption of serverless computing as a viable strategy to enhance development efficiency and reduce operational costs. Future research will delve deeper into the challenges of serverless adoption and explore its applications across various domains, further expanding the knowledge base and practical guidelines for leveraging this promising technology.

FUNDING

The authors would like to acknowledge the support of Prince Sultan University for paying the article processing charges of this publication.

CONFLICTS OF INTEREST

We wish to confirm that there are no known conflicts of interest associated with this publication and there has been no significant financial support for this work that could have influenced its outcome.

PROCESS DATES

12, 2024

This manuscript was initially received for consideration for the journal on 09/30/2024, revisions were received for the manuscript following the double-anonymized peer review on 12/01/2024, the manuscript was formally accepted on 12/01/2024, and the manuscript was finalized for publication on 12/11/2024

CORRESPONDING AUTHOR

Correspondence should be addressed to Mamdouh Alenezi; malenezi@psu.edu.sa

REFERENCES

Achar, S. (2021). Enterprise SaaS workloads on new-generation infrastructure-as-code (IAC) on multicloud platforms. *Global Disclosure of Economics and Business*, *10*(2), 55–74. DOI: 10.18034/gdeb.v10i2.652

Aditya, P., Akkus, I. E., Beck, A., Chen, R., Hilt, V., Rimac, I., Satzke, K., & Stein, M. (2019). Will serverless computing revolutionize NFV? *Proceedings of the IEEE*, *107*(4), 667–678. DOI: 10.1109/JPROC.2019.2898101

Amazon Web Services Lambda – Case Studies — aws.amazon.com [[Accessed 16-11-2023]]. (2022).

Castro, P., Ishakian, V., Muthusamy, V., & Slominski, A. (2019). The server is dead, long live the server: Rise of serverless computing, overview of current state and future trends in research and industry. *arXiv preprint arXiv:1906.02888*.

Chapin, J., & Roberts, M. (2020). *Programming AWS Lambda: Build and deploy serverless applications with java*. O'Reilly Media.

Chaudhry, S. R., Palade, A., Kazmi, A., & Clarke, S. (2020). Improved GOS at the edge using serverless computing to deploy virtual network functions. *IEEE Internet of Things Journal*, *7*(10), 10673–10683. DOI: 10.1109/JIOT.2020.3011057

Chawla, R. (2021). Information flow control for serverless systems. *International Journal of Advanced Computer Science and Applications*, *12*(9). Advance online publication. DOI: 10.14569/IJACSA.2021.0120901

Chowhan, K. (2018). *Hands-on serverless computing: Build, run and orchestrate serverless applications using AWS Lambda, Microsoft Azure functions, and Google Cloud functions*. Packt Publishing Ltd.

Eismann, S., Scheuner, J., Van Eyk, E., Schwinger, M., Grohmann, J., Herbst, N., Abad, C. L., & Iosup, A. (2021). The state of serverless applications: Collection, characterization, and community consensus. *IEEE Transactions on Software Engineering*, *48*(10), 4152–4166. DOI: 10.1109/TSE.2021.3113940

Ellingwood, J. (n.d.). *Traditional vs serverless database architecture comparisons.* Prisma's Data Guide. https://www.prisma.io/dataguide/serverless/traditional-vs-serverless-databases

Fylaktopoulos, G., Goumas, G., Skolarikis, M., Sotiropoulos, A., & Maglogiannis, I. (2016). An overview of platforms for cloud based development. *SpringerPlus*, *5*(1), 1–13. DOI: 10.1186/s40064-016-1688-5 PMID: 26835220

Gabbrielli, M., Giallorenzo, S., Lanese, I., Montesi, F., Peressotti, M., & Zingaro, S. P. (2019). No more, no less: A formal model for serverless computing. *Coordination Models and Languages: 21st IFIP WG 6.1 International Conference, COORDINATION 2019, Held as Part of the 14th International Federated Conference on Distributed Computing Techniques, DisCoTec 2019* (pp. 148–157).

Graduated and incubating projects. (n.d.). Cloud Native Computing Foundation. https://www.cncf.io/projects/

Hassan, H. B., Barakat, S. A., & Sarhan, Q. I. (2021). Survey on serverless computing. *Journal of Cloud Computing (Heidelberg, Germany)*, *10*(1), 1–29. DOI: 10.1186/s13677-021-00253-7

Jain, A., Baarzi, A. F., Kesidis, G., Urgaonkar, B., Alfares, N., & Kandemir, M. (2020). Splitserve: Efficiently splitting Apache spark jobs across FaaS and IaaS. *Proceedings of the 21st International Middleware Conference* (pp. 236–250). DOI: 10.1145/3423211.3425695

Jin, Z., Zhu, Y., Zhu, J., Yu, D., Li, C., Chen, R., Akkus, I. E., & Xu, Y. (2021). Lessons learned from migrating complex stateful applications onto serverless platforms. *Proceedings of the 12th ACM SIGOPS Asia-Pacific Workshop on Systems* (pp. 89–96). DOI: 10.1145/3476886.3477510

Khan, I., Sadad, A., Ali, G., ElAffendi, M., Khan, R., & Sadad, T. (2023). Npr-lbn: Next point of interest recommendation using large bipartite networks with edge and cloud computing. *Journal of Cloud Computing (Heidelberg, Germany)*, *12*(1), 54. DOI: 10.1186/s13677-023-00427-5

Kumari, A., Sahoo, B., Behera, R. K., Misra, S., & Sharma, M. M. (2021). Evaluation of integrated frameworks for optimizing QOS in serverless computing. *Computational Science and Its Applications–ICCSA 2021: 21st International Conference, Cagliari, Italy, September 13–16, 2021. Proceedings*, *21*(Part VII), 277–288.

L'opez, P. G., Arjona, A., Samp'e, J., Slominski, A., & Villard, L. (2020). Triggerflow: Trigger-based orchestration of serverless workflows. *Proceedings of the 14th ACM international conference on distributed and event-based systems* (pp. 3–14). DOI: 10.1145/3401025.3401731

Li, Y., Lin, Y., Wang, Y., Ye, K., & Xu, C. (2022). Serverless computing: State-of-the-art, challenges and opportunities. *IEEE Transactions on Services Computing*, *16*(2), 1522–1539. DOI: 10.1109/TSC.2022.3166553

Liu, F., & Niu, Y. (2023). Demystifying the cost of serverless computing: Towards a win-win deal. *IEEE Transactions on Parallel and Distributed Systems*.

Liu, X., Wen, J., Chen, Z., Li, D., Chen, J., Liu, Y., Wang, H., & Jin, X. (2023). FaaSlight: General application-level cold-start latency optimization for function-as-a-service in serverless computing. *ACM Transactions on Software Engineering and Methodology*, *32*(5), 1–29. DOI: 10.1145/3585007

Mampage, A., Karunasekera, S., & Buyya, R. (2022). A holistic view on resource management in serverless computing environments: Taxonomy and future directions. *ACM Computing Surveys*, *54*(11s), 1–36. DOI: 10.1145/3510412

Mezzalira, L., Hyatt, L., Denti, V., & Jaupaj, Z. (2022, May 4). *Let's architect! Serverless architecture on AWS.* AWS. https://aws.amazon.com/blogs/architecture/lets-architect-serverless-architecture-on-aws/

Nastic, S. (2024). Self-provisioning infrastructures for the next generation serverless computing. *SN Computer Science*, *5*(6), 678. DOI: 10.1007/s42979-024-03022-w

Netflix & AWS Lambda case study. (2014). Amazon Web Services. https://aws.amazon.com/solutions/case-studies/netflix-and-aws-lambda/

Optimizing enterprise economics with serverless architectures. (2021, September 15). AWS. https://docs.aws.amazon.com/whitepapers/latest/optimizing-enterprise-economics-with-serverless/optimizing-enterprise-economics-with-serverless.html

P'erez, A., Molt'o, G., Caballer, M., & Calatrava, A. (2018). Serverless computing for container-based architectures. *Future Generation Computer Systems*, *83*, 50–59. DOI: 10.1016/j.future.2018.01.022

Pu, Q., Venkataraman, S., & Stoica, I. (2019). Shuffling, fast and slow: Scalable analytics on serverless infrastructure. *16th USENIX symposium on networked systems design and implementation (NSDI 19)* (pp. 193–206).

Rajan, R. A. P. (2018). Serverless architecture-a revolution in cloud computing. *2018 Tenth International Conference on Advanced Computing (ICoAC)* (pp. 88–93). DOI: 10.1109/ICoAC44903.2018.8939081

Rehemagi, T. (2020, July 16). *Serverless case study: Coca-Cola.* Dashbird. https://dashbird.io/blog/serverless-case-study-coca-cola/

Retter, M. (2020, July 30). *Serverless case study: Netflix*. Dashbird. https://dashbird.io/blog/serverless-case-study-netflix/

Saba, T., Rehman, A., Haseeb, K., Alam, T., & Jeon, G. (2023). Cloud-edge load balancing distributed protocol for IOE services using swarm intelligence. *Cluster Computing*, *26*(5), 2921–2931. DOI: 10.1007/s10586-022-03916-5 PMID: 36624887

Satzke, K., Akkus, I. E., Chen, R., Rimac, I., Stein, M., Beck, A., Aditya, P., Vanga, M., & Hilt, V. (2020). Efficient GPU sharing for serverless workflows. *Proceedings of the 1st Workshop on High Performance Serverless Computing* (pp. 17–24).

Scheuner, J., & Leitner, P. (2019). Transpiling applications into optimized serverless orchestrations. *2019 IEEE 4th International Workshops on Foundations and Applications of Self* Systems (FAS* W)*, 72–73.

Serverless vs. traditional Architecture: What are the differences? (2019, March 14). Alibaba Cloud. alibabacloud.com

Sewak, M., & Singh, S. (2018). Winning in the era of serverless computing and function as a service. *2018 3rd International Conference for Convergence in Technology (I2CT)*, 1–5.

Shafiei, H., Khonsari, A., & Mousavi, P. (2022). Serverless computing: A survey of opportunities, challenges, and applications. *ACM Computing Surveys*, *54*(11s), 1–32. DOI: 10.1145/3510611

Shahrad, M., Balkind, J., & Wentzlaff, D. (2019). Architectural implications of function-as-a-service computing. *Proceedings of the 52nd annual IEEE/ACM international symposium on microarchitecture*, 1063–1075. DOI: 10.1145/3352460.3358296

Shahrad, M., Fonseca, R., Goiri, I., Chaudhry, G., Batum, P., Cooke, J., Laureano, E., Tresness, C., Russinovich, M., & Bianchini, R. (2020). Serverless in the wild: Characterizing and optimizing the serverless workload at a large cloud provider. *2020 USENIX annual technical conference (USENIX ATC 20)*, 205–218.

Wang, A., Zhang, J., Ma, X., Anwar, A., Rupprecht, L., Skourtis, D., Tarasov, V., Yan, F., & Cheng, Y. (2020). {Infinicache}: Exploiting ephemeral serverless functions to build a {cost-effective} memory cache. *18th USENIX conference on file and storage technologies (FAST 20)*, 267–281.

Wen, J., Chen, Z., Jin, X., & Liu, X. (2023). Rise of the planet of serverless computing: A systematic review. *ACM Transactions on Software Engineering and Methodology*, *32*(5), 1–61. DOI: 10.1145/3579643

Wen, J., Chen, Z., & Liu, X. (2022). A literature review on serverless computing. *arXiv e-prints*, arXiv– 2206.

Wen, J., Chen, Z., Liu, Y., Lou, Y., Ma, Y., Huang, G., Jin, X., & Liu, X. (2021). An empirical study on challenges of application development in serverless computing. *Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering* (pp. 416–428). DOI: 10.1145/3468264.3468558

Why use serverless computing? Pros and cons of serverless. (n.d.). Cloudfare. https://www.cloudflare.com/learning/serverless/why-use-serverless/

Woods, E., Erder, M., & Pureur, P. (2021). *Continuous architecture in practice: Software architecture in the age of agility and devops*. Addison-Wesley Professional.

Ye, W., Khan, A. I., & Kendall, E. A. (2003). Distributed network file storage for a serverless (p2p) network. *The 11th IEEE International Conference on Networks* (pp. 343–347).

Dr. Mamdouh Alenezi is currently the Chairman of the Computer Science department at Prince Sultan University. Dr. Alenezi received his MS and PhD degrees from DePaul University and North Dakota State University in 2011 and 2014, respectively. He has extensive experience in data mining and machine learning where he applied several data mining techniques to solve several Software Engineering problems. He conducted several research and development of predictive models using machine learning to predict fault-prone classes, comprehend source code, and predict the appropriate developer to be assigned to a new bug.