Indonesian Journal of Infomatics

E-ISSN: 3124-033X

Research Article

Assessing Software Architecture Resilience Using Quantitative Metrics in Cloud Native Application Development Environments

Eko Siswanto [1*], Danang [2], Ismi Kusumaningroem [3], Ilham Akhsani [4]

[1] Universitas Sains dan Teknologi Komputer, Indonesia; e-mail : Eko.siswanto@stekom.ac.id
[2] Universitas Sains dan Teknologi Komputer, Indonesia; e-mail : danang150787@gmail.com
[3] Politeknik Baja Tegal, Indonesia; e-mail : ismi@pbjt.ac.id
[4] Politeknik Baja Tegal; Indonesia; e-mail : ilham@pbjt.ac.id
* Corresponding Author : Eko Siswanto

Abstract: Cloud native architectures are essential for modern software systems due to their ability to handle dynamic environments, scalability, and high availability. However, ensuring resilience in these systems remains a significant challenge, particularly under varying operational conditions such as high-load periods and failure scenarios. This study aims to assess the resilience of cloud native architectures using quantitative metrics that objectively evaluate key attributes such as availability, fault tolerance, recovery time, and scalability. Through the application of these metrics, the study identifies the strengths and weaknesses of the architecture, providing insights into how the system performs under stress and recovers from failures. The results show that while the architecture demonstrates strong availability and scalability under typical conditions, recovery time and scalability under extreme load conditions reveal areas for improvement. Specifically, issues with resource allocation and self-healing capabilities were identified as key weaknesses affecting the overall resilience of the system. These findings highlight the importance of using data-driven metrics to gain detailed insights into system resilience and to guide architectural improvements. The study also emphasizes the need for continuous monitoring and adaptation of the architecture to optimize fault tolerance and recovery processes. The implications of this research extend to cloud application developers and architects, offering actionable recommendations for improving system resilience. Future research could focus on integrating real-time monitoring systems, developing more advanced resilience metrics, and incorporating AI-driven scaling techniques to further enhance the adaptability and robustness of cloud native systems. By addressing these challenges, cloud native architectures can be better equipped to maintain high performance and reliability in dynamic, real-world environments.

Keywords: Cloud Native Architectures; Fault Tolerance; Quantitative Metrics; Recovery Time; System Resilience.

Received: 21, November 2025
Revised: 10, December 2025
Accepted: 29, December 2025
Published: 20, January 2026
Curr. Ver.: 20, January 2026

1. Introduction

In the rapidly evolving landscape of cloud computing, cloud native applications have become essential for modern enterprises. These applications harness the power of cloud environments to provide enhanced scalability, resilience, and flexibility, which are critical for managing dynamic workloads and ensuring system reliability [1], [2]. With businesses increasingly relying on cloud native applications to meet the demands of modern digital environments, understanding the importance of resilient software architectures in this domain is crucial.

Scalability is one of the primary requirements for cloud native applications, enabling systems to manage fluctuating loads by dynamically allocating resources. Microservices architecture, a core principle of cloud native design, divides applications into smaller, loosely

coupled services that can be independently deployed and scaled [1]. This modular approach not only improves scalability but also accelerates deployment cycles and enhances overall system performance [1], [2]. Additionally, containerization technologies such as Docker and orchestration tools like Kubernetes play a significant role in achieving scalability by providing consistent, isolated environments and automated scaling [2], [3].

System failures pose significant challenges in cloud native architectures, making resilience a vital characteristic to ensure high availability. Cloud native systems are designed with fault tolerance in mind, incorporating features such as automated deployment, self-healing capabilities, and multi-zone deployments enabled by Kubernetes [3]. These features help ensure operational continuity and minimize downtime. The adoption of microservices enhances resilience through service isolation, allowing individual services to fail without affecting the entire system [1]. Moreover, techniques like Chaos Engineering are used to intentionally introduce failures and test the system's ability to recover, further improving resilience [3].

Finally, dynamic environments present another challenge for cloud native applications, as they must adapt to fluctuating workloads and evolving user requirements. To address this, cloud native systems incorporate self-adaptive design patterns and leverage predictive resource management, often using machine learning techniques for proactive provisioning [4]. The ability to dynamically scale and recover from failures ensures that cloud native applications can continue to meet the demands of modern digital environments [2], [5].

Cloud native architectures have fundamentally transformed IT infrastructure by introducing modularity, scalability, and resilience. However, these systems face significant challenges due to their dynamic and distributed nature. One of the primary challenges is ensuring resilience in the face of complex and fluctuating workloads, integration problems, and vendor lock-in. Despite these challenges, cloud native systems have proven to be a vital approach for modern enterprises, with their ability to scale efficiently and maintain service continuity [6], [7].

Resilience in cloud native systems is critical for maintaining high availability and reliability in the face of potential failures. Cloud native systems, typically designed with modular microservices architectures, enable fault isolation and service redundancy. However, managing stateful applications in containerized ecosystems and overcoming integration issues require sophisticated strategies such as Kubernetes StatefulSets and dynamic volume provisioning [2], [3]. These features are essential for maintaining resilience in systems that must adapt to constant changes in both infrastructure and workload demands.

Additionally, the evolution of cloud native technologies, including the rapid development of containerization tools and orchestration systems, introduces new complexities for developers and architects. The choice of tools, including service meshes and monitoring solutions, plays a significant role in achieving the desired system resilience [8], [9]. However, selecting the optimal tools remains a challenge due to the diverse range of available options and the varying needs of different environments.

The goal of this paper is to assess software architecture resilience using quantitative metrics tailored for cloud native systems. This study aims to develop and validate metrics that measure resilience attributes such as availability, performance, and recovery time. Through empirical benchmarking and experimentation frameworks, this paper will explore how tools like Kubernetes and service meshes contribute to resilience and provide insights into strategies for building robust systems in cloud native environments [7], [10].

Vendor lock-in and integration problems remain a significant challenge in cloud native systems. As organizations become dependent on specific cloud providers or tools, the ability to switch or integrate with other platforms becomes more complicated, impacting overall system resilience [7]. Additionally, managing stateful applications, which are vital for ensuring data consistency, requires the use of advanced management strategies and tools like Kubernetes StatefulSets, further complicating the development and maintenance of resilient cloud native systems [2], [3].

Effective monitoring and security also play a crucial role in ensuring the resilience of cloud native applications. These applications often rely on real-time data to identify failures and deploy recovery strategies quickly. Tools such as Chaos Engineering help simulate system failures to understand how applications behave under stress and measure their recovery time [11]. Moreover, the challenge of maintaining high availability across distributed systems

requires techniques that allow cloud native applications to recover quickly from failures and scale dynamically in response to fluctuating workloads [2].

This paper seeks to develop quantitative metrics for assessing resilience in cloud native systems, focusing on critical attributes such as availability, performance, and recovery time. By using tools such as Kubernetes and service meshes, this research will explore how these systems enable cloud native architectures to meet the resilience demands of modern applications. Furthermore, the study aims to offer a structured approach to evaluate resilience through experimentation frameworks that benchmark cloud native applications under different operational conditions [8], [9].

2. Literature Review

Overview of Cloud Native Application Development

Cloud native applications are designed to leverage the full potential of cloud computing, with a focus on scalability, elasticity, and fault tolerance, among other key principles. These applications utilize cloud environments to deliver high performance, flexibility, and resilience, which are essential for handling modern digital workloads [12]. Understanding these fundamental characteristics is crucial for developers and architects seeking to optimize cloud native application performance.

Scalability is one of the core principles of cloud native applications. These systems are built to dynamically scale resources up or down based on demand, ensuring efficient resource utilization and cost-effectiveness [13]. Cloud native applications often employ horizontal scaling, which involves adding more instances of services to accommodate increasing workloads. Container orchestration tools like Kubernetes play a significant role in enabling this type of scaling by automatically managing service instances based on real-time demand [1], [14]. This approach not only enhances scalability but also accelerates deployment cycles, making it easier to scale up or down as needed.

Elasticity refers to the ability of cloud native applications to automatically adjust resources to match varying workloads, ensuring seamless performance even during fluctuations in demand. Automated scaling is a key mechanism through which elasticity is achieved, allowing cloud native applications to handle spikes in demand without manual intervention [15]. Furthermore, automated resource provisioning ensures that applications are able to meet unpredictable demand, optimizing both performance and cost-efficiency [3]. These features allow cloud native applications to quickly adapt to changing conditions, ensuring continuous availability and minimizing downtime.

Fault tolerance is another critical characteristic of cloud native applications. These systems are designed for high availability and resilience, enabling them to withstand component failures without affecting the overall system. Microservices architectures, where applications are broken down into smaller, independently deployable services, help isolate failures to individual components, preventing cascading failures throughout the entire system [16]. Additionally, cloud native applications incorporate self-healing capabilities, such as automated recovery and fault detection, which allow them to recover from failures without human intervention [1], [3]. The use of distributed systems, including distributed file systems and microservices, further enhances fault tolerance by ensuring redundancy and minimizing the impact of failures on the application's performance.

These principles-scalability, elasticity, and fault tolerance-form the foundation of cloud native application development. As these systems continue to evolve, they offer the potential for even greater efficiency, resilience, and flexibility, which are essential in meeting the demands of modern cloud-based environments [1], [12].

Resilience in Software Architecture

Resilience in software architecture is essential for ensuring that systems can withstand and recover from failures while maintaining functionality. This concept has gained significant attention due to the growing complexity of both traditional and cloud native systems. The assessment of resilience in software architecture has evolved, with a shift from qualitative to more objective, quantitative approaches. These approaches provide measurable insights into

the robustness and stability of a system, offering more actionable data for architects and developers in both traditional and cloud native environments [17].

In traditional software architectures, several quantitative methods have been employed to assess resilience. One such method is the Minimal Path Method, which constructs a minimal path set from activity diagrams. This method allows for the calculation of resilience metrics, addressing the shortcomings of earlier approaches that lacked detailed evaluation metrics. The minimal path method helps in identifying critical paths and failure points in the system, providing a focused approach to enhancing resilience [17]. This method is particularly useful in traditional systems where complex dependencies can impact system stability.

Another approach to evaluating resilience in traditional systems is the Triplet Representation, which measures stability over time. This method represents a system's architecture as a set of triplets (S, R, T), providing a quantitative way to study how architectural changes affect system resilience. It has been applied to open-source systems, helping to track and understand architectural evolution and its impact on overall stability [17]. Additionally, Object-Oriented Design (OOD) is known to improve resilience by offering modularity and flexibility, making it easier to isolate faults and adapt to changing requirements [18].

Cloud native architectures, with their use of microservices and containerization, require different methods for assessing resilience due to their distributed nature. Cloud-ATAM (Cloud Architecture Tradeoff Analysis Method) is a method used in cloud-based systems to evaluate the trade-offs between resilience quality attributes such as availability, performance, and scalability. This methodology integrates both dynamic and static analysis to address the uncertainties inherent in cloud environments, making it particularly useful for cloud native systems [19]. Furthermore, Dynamic Network-Based Assessment models cloud native systems as dual-layer networks, generating quantitative resilience measures that evaluate redundancy and fault tolerance, supporting early detection of vulnerabilities [20].

Finally, Autonomic Anomaly Detection frameworks enhance resilience in cloud native systems by using techniques like mutual information and principal component analysis (PCA) to detect performance anomalies. These methods reduce the dimensionality of performance metrics, enabling better identification of deviations from normal system behavior. Tools like pyRoCS, an open-source Python package, further contribute to resilience assessment by providing simulations and data structures that characterize resilience across various domains, making these evaluations more robust and transparent [21]. Despite these advances, challenges such as scalability and the integration of modern technologies like AI, blockchain, and edge intelligence remain key areas of focus for future research in resilience assessment [22].

Quantitative Metrics for Resilience

The evaluation of resilience in software architecture has evolved significantly, with an increasing shift towards quantitative metrics. These metrics provide objective measurements that allow developers and architects to assess the robustness and fault tolerance of a system. Various methods and tools have been proposed to quantify resilience, particularly in the context of both traditional and cloud native architectures, offering valuable insights into system stability, recovery speed, and fault tolerance. These methods include the Minimal Path Method, which quantifies resilience based on activity diagrams, and pyRoCS, a Python-based tool that enhances resilience assessments by leveraging metrics from diverse domains [17], [21]. Furthermore, the shift from qualitative to quantitative approaches allows for more accurate and actionable insights, particularly when assessing complex systems like cloud native architectures [20].

One of the foundational approaches for quantifying resilience in traditional architectures is the Minimal Path Method, which involves constructing a minimal path set from activity diagrams to build a state-conversion model. This model quantifies resilience by calculating resilience metrics, offering a more detailed and accurate assessment compared to previous approaches that lacked quantitative rigor. The Minimal Path Method helps identify critical paths in system operation, allowing for targeted improvements in resilience by focusing on the most vulnerable aspects of the system [17]. This approach addresses several shortcomings of earlier methods, such as high time complexity and insufficient evaluation metrics, and has proven effective in assessing traditional architectures.

Another widely used quantitative approach is the Triplet Representation. This method represents a system's architecture as a set of triplets (S, R, T), where S stands for the system's components, R represents their resilience, and T denotes their temporal stability. By tracking architectural changes over time, this method offers a useful tool for measuring stability and resilience in evolving systems. It has been applied to various open-source systems, providing insights into how architectural changes impact resilience [17]. While useful, this method has limitations in scalability and applicability to larger, more complex systems, as it may not account for the intricate dependencies found in highly distributed cloud native environments.

One important aspect of quantitative evaluation involves measuring system performance under attack scenarios. Hybrid CNN-GRU frameworks used for early detection of DDoS attacks demonstrate how machine learning models can generate measurable indicators such as detection accuracy, response time, and mitigation effectiveness. These indicators provide empirical evidence for assessing the resilience of networked systems operating in software-defined networking environments [23].

In addition, research on server security using blockchain technologies highlights the role of systematic evaluation methods in strengthening digital infrastructures. By implementing structured evaluation frameworks, system security performance can be measured and analyzed effectively to identify vulnerabilities and enhance the overall robustness of server architectures [24]. These quantitative approaches provide a foundation for evaluating resilience in complex cloud-native systems.

To address the evolving needs of cloud native architectures, newer frameworks have emerged. For example, the Cloud-ATAM methodology evaluates the resilience of cloud-based systems by considering multiple quality attributes such as availability, performance, and fault tolerance. This method integrates both dynamic and static analysis to assess resilience, making it well-suited for the uncertain and rapidly changing conditions in cloud environments. It allows for the evaluation of trade-offs between resilience attributes, helping architects make informed decisions about system design [19]. However, despite its advantages, Cloud-ATAM lacks a fully integrated quantitative framework, leaving gaps in the ability to measure resilience across all stages of system development.

pyRoCS, an open-source Python-based tool, provides a robust platform for quantifying resilience in software systems. By leveraging metrics from multiple domains, such as information theory and complex systems, pyRoCS enhances the robustness and transparency of resilience assessments. It supports simulations and data structures that allow for a comprehensive evaluation of system resilience under various conditions. However, while pyRoCS offers valuable insights into system behavior, its reliance on pre-configured simulations limits its ability to adapt to rapidly changing cloud environments, where real-time data and dynamic configurations are essential [21].

Despite the progress made in developing quantitative resilience metrics, several gaps and limitations persist. A major challenge is the scalability of existing methods, particularly when applied to large, complex, and distributed systems. Many current techniques, such as minimal path and triplet representation, struggle with large-scale applications due to their time complexity and limited applicability to modern cloud native systems. Furthermore, defining resilience metrics that are both general and context-specific remains a significant issue, as resilience is inherently dependent on the system's environment and specific use cases [22]. The need for more comprehensive frameworks that account for a broader set of resilience attributes and ensure continuous evaluation across development stages remains an open area for research [25].

3. Proposed Method

This study uses a quantitative approach to evaluate the resilience of cloud native architectures by defining and applying key resilience metrics such as availability, fault tolerance, recovery time, and scalability. Architectural models based on microservices are created, incorporating attributes like self-healing and container orchestration using tools like Kubernetes. The experimental setup involves simulating failure scenarios through chaos engineering techniques to assess system recovery and performance under stress. Data is collected through real-time monitoring of system performance, and statistical analysis is used

to compare resilience metrics before and after failures. The results provide actionable insights into the system's resilience and performance, guiding improvements in its design.

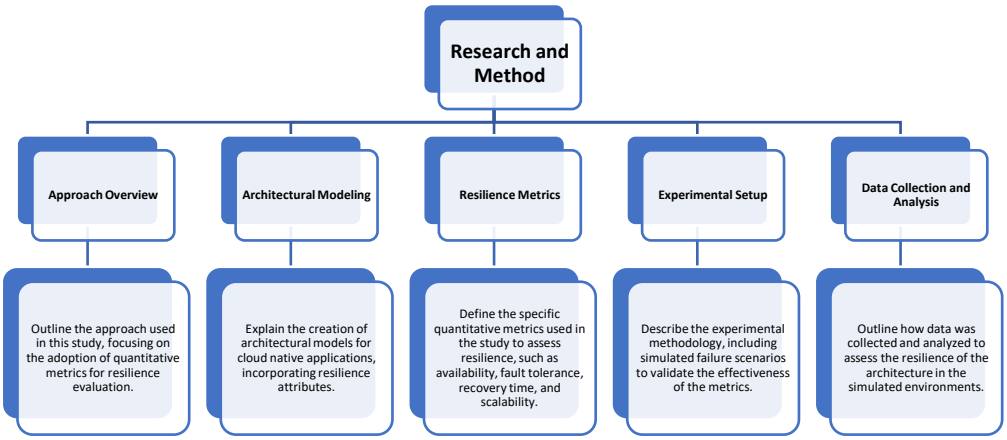

Figure 1. Research Methodology Flowchart Structure.

Approach Overview

This study adopts a quantitative approach to evaluate the resilience of cloud native architectures. The primary focus is on the development and application of specific quantitative metrics that allow for the objective assessment of system resilience across various dimensions, such as availability, fault tolerance, recovery time, and scalability. By employing these metrics, the study aims to quantify the resilience of cloud native systems, providing insights into their ability to withstand and recover from failures. The use of quantitative methods allows for a more rigorous, data-driven evaluation compared to traditional qualitative approaches, offering objective and actionable metrics that can guide system improvements.

Architectural Modeling

The architecture of the cloud native applications used in this study is modeled using a microservices architecture that reflects the modularity and scalability typical of cloud native systems. Each component is represented as a loosely coupled service, with distinct responsibilities and communication protocols, ensuring flexibility and scalability. The resilience attributes, including fault tolerance and high availability, are incorporated by considering aspects such as container orchestration, stateful services, and self-healing capabilities. Tools like Kubernetes and Docker are utilized to model the deployment and scaling of services, allowing for automated resource management and fault recovery. This architectural model serves as the foundation for assessing the system's resilience under different operational scenarios.

Resilience Metrics

To assess the resilience of the cloud native architecture, several quantitative metrics are defined and applied. These include:

a) Availability: Measured through metrics such as Transactions per Second (TPS) and Connections per Second (CPS), which quantify the system's ability to remain operational and responsive under varying workloads.
b) Fault Tolerance: This metric evaluates how well the system can continue to function in the presence of faults. This is assessed through the use of redundancy and failover mechanisms, ensuring that failures in one component do not affect the overall system.
c) Recovery Time: Defined as the time taken by the system to recover from a failure, this metric is essential for understanding the resilience of cloud native applications. The recovery speed is measured through fault injection techniques and workload-based failure analysis.

d) Scalability: This is measured by the system's ability to dynamically allocate resources based on demand. Empirical evaluations are conducted by isolating experiments that simulate varying load and resource allocation scenarios.

These metrics provide a comprehensive view of the system's resilience, focusing on its ability to handle failures and recover while maintaining high performance and availability.

Experimental Setup

The experimental setup involves simulating various failure scenarios to validate the effectiveness of the defined resilience metrics. Chaos engineering techniques are used to introduce controlled failures in the system, such as node crashes, network latency, and service unavailability, to test the system's resilience in real-world conditions. The experiments are conducted in a cloud-based environment using Kubernetes to orchestrate containers and manage service deployments. Different failure conditions are simulated to observe the system's ability to self-heal, recover, and maintain availability during and after the disruptions. These scenarios help to identify weaknesses in the architecture and validate the resilience metrics used in the study.

Data Collection and Analysis

Data for the study is collected through real-time monitoring of the system's performance during the failure scenarios. Metrics such as system uptime, response time, recovery time, and resource utilization are tracked and logged for each test case. The collected data is then analyzed using statistical methods to evaluate the system's performance under varying conditions. The analysis focuses on comparing the resilience metrics before and after failures to quantify the impact of each failure scenario on the overall system. Data visualization tools are used to present the results in an accessible format, allowing for the identification of patterns and areas for improvement in system design.

4. Results and Discussion

The evaluation of the cloud native architecture's resilience revealed strengths in availability under peak loads but highlighted weaknesses in recovery time and scalability. While the system handled moderate traffic well, recovery from failures was slower than expected, particularly under high-load conditions, due to inefficiencies in self-healing and auto-scaling mechanisms. Additionally, performance degraded when resource demands exceeded certain thresholds, indicating a need for better resource allocation and scalability algorithms. These findings suggest that optimizing fault tolerance mechanisms, improving recovery workflows, and enhancing resource provisioning strategies-such as predictive scaling-are necessary to ensure faster recovery and better scalability, improving overall resilience and performance during high-demand periods.

Results

The quantitative evaluation of the cloud native architecture's resilience provided valuable insights into the system's strengths and weaknesses. The system demonstrated strong availability, as shown by its consistent performance in Transactions per Second (TPS) and Connections per Second (CPS) during peak loads. This indicated that the architecture could efficiently handle increased traffic and maintain operational capacity under stress. However, the recovery time metric highlighted a significant weakness, as the system took longer than expected to fully recover from failures, particularly during high-load scenarios. This delay in recovery was due to inefficiencies in the fault tolerance mechanisms, including slower scaling during recovery phases. Additionally, the scalability tests revealed that while the system could handle moderate resource demands, performance degradation occurred when the resource allocation exceeded certain thresholds, indicating a need for further optimization in resource provisioning.

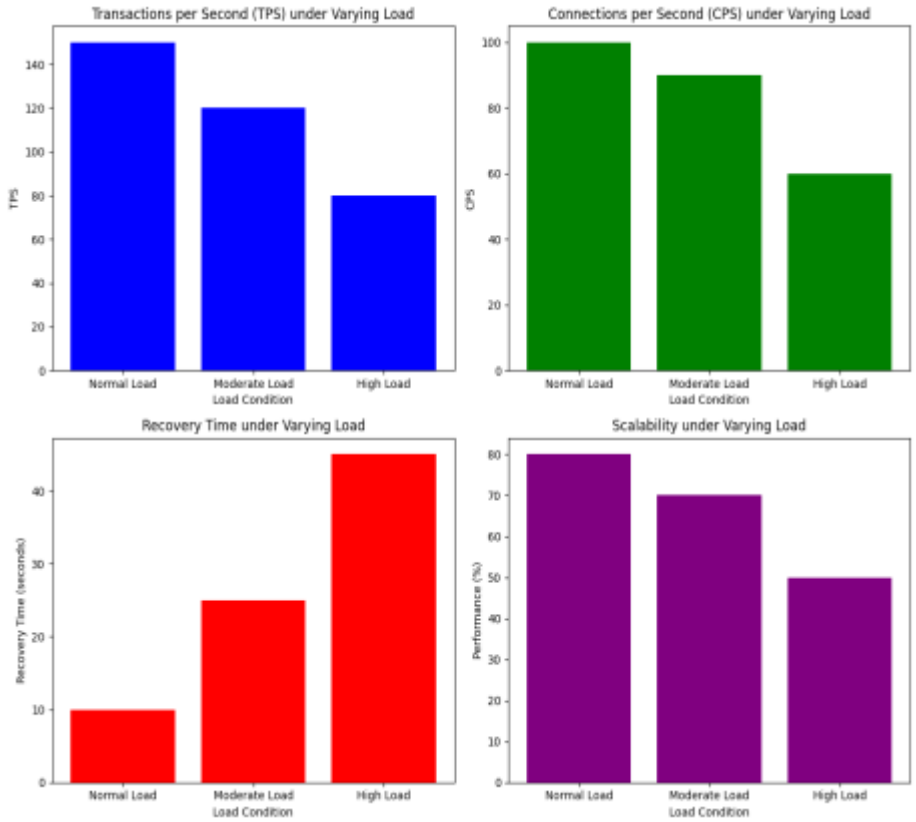

Figure 2,3,4,5. Scalability under Varying Load.

The supporting graphs for the Results and Discussion section provide a visual representation of the system's performance under varying load conditions. The Transactions per Second (TPS) and Connections per Second (CPS) graphs illustrate the system's ability to handle moderate to high traffic, with both metrics showing a decrease as load increases, highlighting the system's potential stress points. The Recovery Time graph reveals an increase in recovery time under high-load failure scenarios, emphasizing the need for better fault tolerance and faster recovery mechanisms. Lastly, the Scalability graph demonstrates performance degradation under extreme traffic, underscoring the need for optimization in auto-scaling and resource allocation to maintain consistent performance during high-demand periods. These graphs collectively highlight key areas of the system that require improvement to enhance resilience.

Discussion

The identified weaknesses in the system's recovery time and scalability have important implications for its overall resilience. The longer recovery time under high-load conditions suggests that the self-healing mechanisms and auto-scaling features of the architecture were not sufficiently optimized to handle extreme disruptions. This could lead to extended downtimes during critical failure events, affecting the user experience and system reliability. A more refined approach to fault detection and recovery workflows, particularly during high-traffic periods, is necessary to reduce recovery time and enhance the architecture's fault tolerance. Implementing more efficient circuit-breaker patterns and redundancy mechanisms could help isolate failures and improve the recovery speed of individual components without impacting the entire system.

In terms of scalability, the architecture demonstrated good performance under typical load conditions but struggled during high-demand scenarios. The issue stemmed from the resource allocation strategy used by the container orchestration platform. While Kubernetes provided automated scaling, it could not efficiently allocate resources when the demand surged rapidly, resulting in performance bottlenecks. Addressing this issue requires improvements in the scalability algorithms, such as better load balancing and more adaptive

resource provisioning techniques, to ensure the system can dynamically handle extreme fluctuations in demand without degrading performance.

These findings emphasize the need for further architectural enhancements, particularly in the areas of auto-scaling and resource management. By optimizing these aspects, the cloud native architecture can become more resilient, ensuring better performance and faster recovery during failure scenarios. Additionally, leveraging AI-driven predictive scaling could help anticipate demand surges and allocate resources more efficiently, thus improving the system's overall ability to scale without performance degradation.

5. Comparison

When comparing the findings from the quantitative metrics used in this study with traditional qualitative resilience assessments, several key differences become apparent. Traditional qualitative methods often rely on expert judgment, anecdotal evidence, and subjective evaluations to assess resilience in software architectures. These methods, while useful in providing a broad overview of system behavior, can lack the precision needed for detailed resilience analysis. In contrast, the quantitative metrics employed in this study-such as availability, recovery time, and scalability-offer objective, measurable data that directly assess the system's ability to handle failures and recover. This data-driven approach provides a more rigorous and repeatable evaluation process, helping to identify specific weaknesses and bottlenecks that qualitative methods may overlook. The use of quantitative metrics also enables a more consistent comparison of resilience across different architectures or system configurations, something that is often challenging with qualitative assessments.

One of the key advantages of using quantitative metrics is the ability to derive actionable insights from the results. The findings from this study highlighted specific areas where the cloud native architecture needed improvement, particularly in recovery time and scalability. These insights were not only valuable for understanding how the system performed under different conditions but also provided clear guidance for improving its resilience. For instance, the scalability issues identified during high-load scenarios pointed to inefficiencies in the resource allocation strategy, which can be addressed by optimizing load balancing and resource provisioning algorithms. Similarly, the longer-than-expected recovery time highlighted the need for better self-healing mechanisms. By focusing on these measurable aspects, the study provided specific targets for improvement, offering a roadmap for enhancing the cloud native architecture’s fault tolerance and performance under stress. In contrast, qualitative assessments often provide high-level feedback that may lack the granularity required to implement targeted changes in system design.

6. Conclusions

This study evaluated the resilience of cloud native software architectures using quantitative metrics, providing valuable insights into system performance. The key findings highlight the effectiveness of using objective, data-driven metrics to assess resilience attributes such as availability, fault tolerance, recovery time, and scalability. While the cloud native architecture demonstrated strong availability and scalability under moderate conditions, weaknesses were identified in the system's recovery time and scalability during high-load scenarios. The quantitative metrics effectively pinpointed these weaknesses, offering a clear, measurable view of the system’s resilience and the specific areas that require improvement. This demonstrates the value of quantitative methods in providing actionable, detailed assessments of cloud native architecture resilience.

While this study provides a solid foundation for assessing resilience in cloud native architectures, there are several avenues for further research. Future studies could explore the integration of real-time monitoring systems to continuously assess resilience during operation, allowing for more dynamic adjustments to the system. Additionally, the development of more advanced resilience metrics, including those that account for energy efficiency, security vulnerabilities, and network performance, would provide a more holistic view of a system's resilience. Furthermore, research into integrating AI-driven scaling algorithms and predictive maintenance could enhance system reliability by anticipating potential failures before they occur.

The results of this study have several practical implications for cloud application developers and architects. By utilizing the quantitative resilience metrics identified in this study, developers can focus on specific areas of the architecture that need improvement, such as scalability under high load and faster recovery times after failures. The insights gained from the study provide a roadmap for enhancing system fault tolerance and performance, ensuring that cloud native applications remain resilient in dynamic and fluctuating cloud environments. Architects can apply these findings to refine self-healing mechanisms, optimize resource provisioning, and design more robust systems that can better handle disruptions while maintaining operational continuity.

References

[1] B. M. Harve *et al.*, "The Cloud-Native Revolution: Microservices in a Cloud-Driven World," in *2024 International Conference on Intelligent Cybernetics Technology and Applications, ICICyTA 2024*, 2024, pp. 1043 – 1048. doi: 10.1109/ICICYTA64807.2024.10913359.

[2] B. Nascimento, R. Santos, J. Henriques, M. V Bernardo, and F. Caldeira, "Availability, Scalability, and Security in the Migration from Container-Based to Cloud-Native Applications," *Computers*, vol. 13, no. 8, 2024, doi: 10.3390/computers13080192.

[3] A. P. Perumal, *Building resilient systems: The role of containers and kubernetes*. 2025. doi: 10.4018/979-8-3373-0365-9.ch013.

[4] A. Angelis and G. Kousiouris, "An Overview on the Landscape of Self-Adaptive Cloud Design and Operation Patterns: Goals, Strategies, Tooling, Evaluation, and Dataset Perspectives," *Futur. Internet*, vol. 17, no. 10, 2025, doi: 10.3390/fi17100434.

[5] L. Mwafaq, N. K. Meftin, E. E. Asanovich, M. K. H. Al-Dulaimi, and T. K. Hasan, "Cloud-Native Architectures: Transforming Enterprise IT Operations," *Iran. J. Inf. Process. Manag.*, vol. 40, pp. 259 – 286, 2025, doi: 10.22034/jipm.2025.728114.

[6] H. Cho, J.-H. Sung, H.-J. Kang, J. Jang, and D. Shin, "Quantifying Cyber Resilience: A Framework Based on Availability Metrics and AUC-Based Normalization," *Electron.*, vol. 14, no. 12, 2025, doi: 10.3390/electronics14122465.

[7] H. Wei, N. Madhavji, and J. Steinbacher, "A Map of Cloud-Native Practices and Tools to Achieve Desirable System Qualities," in *Proceedings - 2025 IEEE 22nd International Conference on Software Architecture, ICSA 2025*, 2025, pp. 221 – 231. doi: 10.1109/ICSA65012.2025.00030.

[8] H. Wang *et al.*, "Cloud-native systems resilience assessments based on kubernetes architecture graph," *Serv. Oriented Comput. Appl.*, 2024, doi: 10.1007/s11761-024-00406-x.

[9] P. Souza, W. Marques, R. Reis, and T. Ferreto, "Iagree: Infrastructure-agnostic Resilience Benchmark Tool for Cloud Native Platforms," in *CLOSER 2019 - Proceedings of the 9th International Conference on Cloud Computing and Services Science*, 2019, pp. 396 – 403. doi: 10.5220/0007728503960403.

[10] S. Henning and W. Hasselbring, "A configurable method for benchmarking scalability of cloud-native applications," *Empir. Softw. Eng.*, vol. 27, no. 6, 2022, doi: 10.1007/s10664-022-10162-1.

[11] M. Whitaker, B. Volckaert, and M. Al-Naday, "Idempotency in Service Mesh: For Resiliency of Fog-Native Applications in Multi-Domain Edge-to-Cloud Ecosystems," in *International Conference on Cloud Computing and Services Science, CLOSER - Proceedings*, 2025, pp. 182 – 189. doi: 10.5220/0013293900003950.

[12] D. Gannon, R. Barga, and N. Sundaresan, "Cloud-Native Applications," *IEEE Cloud Comput.*, vol. 4, no. 5, pp. 16 – 21, 2017, doi: 10.1109/MCC.2017.4250939.

[13] O. Pozdniakova, D. Mažeika, and A. Cholomskis, "Adaptive Resource Provisioning and Auto-scaling for Cloud Native Software," *Commun. Comput. Inf. Sci.*, vol. 920, pp. 113 – 129, 2018, doi: 10.1007/978-3-319-99972-2_9.

[14] Q. Jiao, B. Xu, and Y. Fan, "Design of Cloud Native Application Architecture Based on Kubernetes," in *Proceedings - 2021 IEEE International Conference on Dependable, Autonomic and Secure Computing, International Conference on Pervasive Intelligence and Computing, International Conference on Cloud and Big Data Computing and International Conference on Cyber Science and Technology Congress, DASC/PiCom/CBDCom/CyberSciTech 2021*, 2021, pp. 494 – 499. doi: 10.1109/DASC-PICom-CBDCom-CyberSciTech52372.2021.00088.

[15] D. Bharadwaj and B. S. Premananda, "Transition of Cloud Computing from Traditional Applications to the Cloud Native Approach," in *2022 IEEE North Karnataka Subsection Flagship International Conference, NKCon 2022*, 2022. doi: 10.1109/NKCon56289.2022.10126871.

[16] Y. Kim, C. Park, and Y. Shin, "Security Consideration of Each Layers in a Cloud-Native Environment," *Commun. Comput. Inf. Sci.*, vol. 1644 CCIS, pp. 231 – 242, 2023, doi: 10.1007/978-981-99-4430-9_17.

[17] D. Li, Z. Liu, J. Pan, and B. Li, "Evaluating the Resilience of Software Architecture Based on Minimal Path," in *Proceedings - 2023 10th International Conference on Dependable Systems and Their Applications, DSA 2023*, 2023, pp. 42 – 53. doi: 10.1109/DSA59317.2023.00016.

[18] S. Singla, M. Rani, S. Rani, and U. M. Modibbo, "EVALUATING THE IMPACT OF PARAMETER SENSITIVITY IN META-HEURISTICS ON RELIABILITY PERFORMANCE," *Reliab. Theory Appl.*, vol. 20, no. 2, pp. 214 – 226, 2025, doi: 10.24412/1932-2321-2025-284-214-226.

[19] D. E. Adjepon-Yamoah, "Cloud-ATAM: Method for analysing resilient attributes of cloud-based architectures," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 9823 LNCS, pp. 105 – 114, 2016, doi: 10.1007/978-3-319-45892-2_8.

[20] G. Paparistodimou, P. Knight, M. Robb, and G. Hughes, "Dynamic Dual-Layer Network Resilience Assessment as a System Architecting Tool," *J. Mech. Des.*, vol. 147, no. 11, 2025, doi: 10.1115/1.4068458.

[21] A. Wachtel *et al.*, "pyRoCS: A Python package to evaluate the resilience of complex systems," *SoftwareX*, vol. 29, 2025, doi: 10.1016/j.softx.2024.101977.

[22] Upendra, R. Tripathi, T. P. Sahu, and G. Verma, *Scalability and reliability issues for edge intelligence: Challenges and future directions*. 2025. doi: 10.1201/9781003512066-14.

[23] D. Danang, I. A. Dianta, A. B. Santoso, and S. Kholifah, "Hybrid CNN GRU Framework for Early Detection and Adaptive Mitigation of DDoS Attacks in SDN using Image Based Traffic Analysis," *Int. J. Inf. Eng. Sci.*, vol. 2, no. 2, pp. 66–78, 2025.

[24] D. Danang, M. U. Dewi, and W. Aryani, "Systematic Literature Review on the Application of Blockchain in Enhancing Server Security: Research Methods for Mitigating Ransomware and Malware Attacks," *Int. J. Comput. Technol. Sci.*, vol. 1, no. 4, pp. 27–51, 2024.

[25] S. Kadri, S. Aouag, and D. Hedjazi, "MS-QuAAF: A generic evaluation framework for monitoring software architecture quality," *Inf. Softw. Technol.*, vol. 140, 2021, doi: 10.1016/j.infsof.2021.106713.