

- ORIGINAL ARTICLE -

Scalability in Microservices: A systematic literature review

Escalabilidad en Microservicios: Una revisión sistemática de la literatura

Nishal Gurung¹ , Sushil Shrestha¹ , and Rajani Chulyadyo¹ 

¹Department of Computer Science and Engineering, Kathmandu University, Nepal
 nishal.gurung4@gmail.com, sushil@ku.edu.np, rajani.chulyadyo@ku.edu.np

Abstract

The scalability of microservices architectures is crucial for modern software systems, yet it presents significant challenges due to their inherent complexities. This study aims to systematically review existing literature on the scalability of microservices, identifying key strategies, challenges, and emerging trends. We conducted a systematic literature review following the PRISMA guidelines, analyzing 55 scholarly articles that specifically address the scalability of microservices. The review focused on various scaling approaches, metrics, and the effectiveness of autoscaling mechanisms. Our findings reveal a diverse body of literature with a predominant focus on autoscaling strategies, particularly those utilizing machine learning. Key challenges identified include accurate metrics collection, dynamic scaling decision-making, and balancing performance with cost and security. While progress has been made in addressing scalability challenges, significant gaps remain, particularly in standardizing autoscaling metrics. Future research should focus on developing robust, adaptive autoscaling systems that can effectively manage real-world complexities and dynamic workloads, ensuring both performance and cost optimization in microservices architectures.

Keywords: Microservices Architecture, Scalability, Scalability Challenges, Autoscaling

Resumen

La escalabilidad de las arquitecturas de microservicios es crucial para los sistemas de software modernos, pero presenta desafíos significativos debido a sus complejidades inherentes. Este estudio tiene como objetivo revisar sistemáticamente la literatura existente sobre la escalabilidad de los microservicios, identificando estrategias clave, desafíos y tendencias emergentes. Llevamos a cabo una revisión sistemática de la literatura siguiendo las directrices PRISMA, analizando 55 artículos académicos que abordan específicamente la escalabilidad de los microservicios. La revisión se centró en diversos enfoques de escalado, métricas y la efectividad de los mecanismos de autoescalado. Nuestros hallazgos revelan un cuerpo diverso de liter-

atura con un enfoque predominante en las estrategias de autoescalado, particularmente aquellas que utilizan aprendizaje automático. Los desafíos clave identificados incluyen la recopilación precisa de métricas, la toma de decisiones de escalado dinámico y el equilibrio entre el rendimiento, el costo y la seguridad. Si bien se han logrado avances en la resolución de los desafíos de escalabilidad, persisten brechas significativas, particularmente en la estandarización de las métricas de autoescalado. Las investigaciones futuras deberían centrarse en desarrollar sistemas de autoescalado robustos y adaptativos que puedan gestionar eficazmente las complejidades del mundo real y las cargas de trabajo dinámicas, garantizando tanto la optimización del rendimiento como del costo en las arquitecturas de microservicios.

Palabras claves: Arquitectura de Microservicios, Escalabilidad, Desafíos de Escalabilidad, Autoescalado

1 Introduction

In recent years, the adoption of microservices architecture has gained significant traction due to its ability to enhance the scalability, agility, and resilience in software systems [1]. Microservices are self-contained and independently deployable services, which makes them more convenient to work with compared to larger monolithic architectures [2]. This architectural style allows organizations to develop and deploy applications more rapidly, adapt to changing business requirements, and scale components independently based on demand.

With the growing popularity of microservices, understanding scalability within this architectural paradigm has become increasingly important. Scalability, the ability of a system to handle increased workload by adding resources or replicating components, is essential for maintaining optimal performance and user experience [3]. However, achieving scalability in microservices introduces unique challenges that differ significantly from traditional monolithic architectures. These challenges stem from the dynamic and heterogeneous nature of microservices environments, which are typically deployed in containerized or virtualized settings where instances can be added or

removed dynamically based on workload fluctuations. Ensuring the scalability of individual microservices while preserving overall system performance and resource efficiency requires sophisticated orchestration and autoscaling mechanisms.

Despite its importance as a key quality attribute (QA) in microservices architectures, scalability has not been comprehensively studied in existing literature. While there are several Systematic Literature Reviews (SLRs) addressing various aspects of microservices, only few directly targets scalability, with the rest offering broader perspectives on QAs, testing, security and migrations.

To bridge these gaps, this SLR aims to provide a comprehensive analysis of scalability in microservices. By synthesizing findings from diverse research studies, it seeks to uncover key concepts, methodologies and challenges related to scalability in this architectural paradigm. The review also endeavors to identify emerging trends and offer actionable insights for researchers, practitioners, and organizations aiming to build scalable microservices systems.

Through this study, we aim to enhance the understanding of scalability in microservices and offer valuable guidance for future research. This paper seeks to address the following research questions:

- RQ1: What types of research have been conducted in the scalability domain of microservices?
- RQ2: What are the major challenges faced by researchers while trying to scale microservices architectures?
- RQ3: What are the most commonly used tools in microservices architectures?

By addressing these questions, this SLR aspires to advance knowledge in this critical area of software engineering and support organizations in their microservices adoption journeys.

The structure of this paper is as follows: Section 2 provides an overview of the related studies. In Section 3, we outline our methodology, while Section 4 presents our findings and addresses the research questions. Our evaluations are discussed in Section 5. Finally, Section 6 concludes the paper and explores potential directions for future research.

2 Related Work

A variety of SLRs and Systematic Mapping Studies (SMSs) have been published across different domains of microservices. Among the key research areas, scalability, especially its growth alongside Machine Learning (ML), has gained significant attention. Therefore, this review comprehensively examines the body of work related to scalability in the microservices domain. In addition, we explore the different types of research in scalability and the challenges faced when

scaling microservices. Table 4 provides a comparative overview of existing studies in this area and highlights how they relate to our work.

Several papers have addressed migration from monolithic architectures to microservices [4, 5]. Similarly, studies have focused on microservices adoption and adaptation [6, 7]. There are also works that explored research trends and the challenges within microservice architecture [8, 9, 10]. Meanwhile, other papers have reviewed testing methods, deployment strategies, and security concerns in microservices [11, 12, 13]. Additionally, papers that discuss architectural design patterns [14, 15] and cover the challenges of microservice architecture along with proposed solutions [16] are also notable.

In the autoscaling domain, [10] reviewed relevant studies, identifying key challenges such as resource prediction accuracy, sudden demand spikes, integration complexities, and scaling overhead. The study examined both proactive and reactive autoscaling methods, emphasizing the need to balance performance, cost, and energy efficiency. Furthermore, the paper highlighted future research directions, including improvements in machine learning-based predictions, hybrid scaling strategies, cost-aware algorithms, and security enhancements.

Other works have also touched on scalability within the broader context of microservices research. For example, an overview of microservices, highlighting their motivations and emerging standards, has been provided [17]. Likewise, the analysis of various quality attributes, such as the interactions between functionality and performance, has been discussed, but without delving deeply into scalability challenges [18].

Despite the considerable number of studies on microservices, including significant discussions on scalability, there remains a noticeable gap in research that thoroughly explores the specific trends and challenges of scalability in microservice architectures. This SLR aims to fill this gap by offering a comprehensive analysis of scalability in microservices.

3 Methods

Our research methodology was based on the Preferred Reporting Items for Systematic Reviews and Meta-Analyses (PRISMA) 2020 statement [22]. PRISMA offers a standardized and transparent approach, ensuring methodological rigor in the review process. It suggests detailing the search strategy, study selection process, data collection methods, risk of bias assessment, and presenting results systematically to ensure clarity and comprehensiveness in the review. By adhering to PRISMA guidelines, we aim to enhance the quality and credibility of our SLR, providing readers with a comprehensive synthesis of existing evidence in our research domain.

Table 1: Comparison with Previous Related Studies

Paper	# of studies	Time span	Focus Areas
[19]	33	2014 - 2016	Architectural Challenges, Representation Methods and QAs
[17]	37	2015 - 2016	Microservices Research Types, Practical Motivations, Emerging Standards and Tools
[14]	42	up to 2016	Microservices Architecture Patterns, Advantages and Disadvantages of Patterns
[8]	38	2016 to early 2018	Microservices Research Trends, Areas of Interest and Research Gaps
[15]	42	2014 - 2016	Microservices Architecture Patterns, Deployment Strategies, Data Storage Patterns, and DevOps Techniques
[11]	15	2013 - 2018	Microservices Testing Issues, Quality Concerns in Testing, Challenges and Solutions
[7]	877	January 2013 - April 2020	Microservices Adoption and Transition, Granularity Modeling Approaches, QAs and Reasoning for Granularity
[18]	72	up to June 2018	QAs in Microservices, Tactics for Improving QAs
[6]	62	up to January 2021	Microservices Adaptation Targets, Application vs. Architecture Level Adaptations Techniques, and Machine Learning in Adaptations
[20]	55	2018 - 2021	Microservice Analysis Methods, Problems and Opportunities in Microservice Analysis, Relationship with Other Architectures, and Future Research Directions in Microservice Analysis
[12]	38	2014 - 2019	Microservices Deployment, Communication Patterns, Issues and Research Directions
[16]	85	January 2014 - February 2022	Microservice Architecture Challenges, and Proposed Solutions
[4]	58	2015 - 2021	Quality-Driven Migration to Microservices, QA in Migration, and Migration Phases
[21]	16	2014 - 2024	Distributed Transaction Management in Microservice Architecture, and Solutions to Transaction Challenges
[9]	44	2016 - 2023	Publishing Trends in Microservices Studies, Predominant Topics, Domains of Implementation and Future Research Directions
[13]	54	up to April 2024	Inter-Service Security in Microservices, Security Threats, Categorization, and Mitigation Strategies
[5]	48	2020 - 2023	QAs in Migration, Impact of Migration on Quality, Comparison of Monolithic and Microservice Quality, Metrics for QAs
[10]	-	-	Auto-Scaling in Cloud Computing, Auto-Scaling Algorithms, Challenges and Solutions in Auto-Scaling
Our Study	55	up to February 2025	Scalability in Microservices, Research Types in Scalability, and Challenges in Scaling Microservices

3.1 Search strategy

For this systematic search, a comprehensive search strategy was thoroughly developed to identify relevant literature. The search strategy was tailored to four prominent databases: Scopus, OpenAlex, Crossref, and Semantic Scholar using Publish or Perish application [23] and is conducted at March 02, 2025. The search terms employed were (*Scalability OR Scalable*) AND *Microservice*.

The search encompassed articles from the inception of each database until February 2025, specifically targeting journal articles, conference and proceedings articles, review papers, and preprint articles published exclusively in English.

3.2 Inclusion and exclusion criteria

Table 2 outlines our inclusion and exclusion criteria. We excluded papers that were in languages other than English, theses, books and keynote talk abstracts.

3.3 Selection process

Figure 1 illustrates the process of retrieving and selecting studies, following the PRISMA 2020 guidelines for systematic review reporting. We applied the inclusion and exclusion criteria to filter the papers through the following steps.

- **Stage 1:** We executed the search query across four databases: Scopus, OpenAlex, Crossref, and Semantic Scholar using Publish or Perish tool, retrieving 5200 papers. After removing 2910 duplicates and ineligible records, 2290 papers remained.

- **Stage 2:** We excluded studies that were non-English, theses, book chapters or keynotes abstracts, narrowing the selection to 94 papers.
- **Stage 3:** The next step involved further screening by reviewing abstracts, resulting in 75 potentially relevant papers.
- **Stage 4:** We then examined the full texts and identified 55 studies that met our criteria.

Ultimately, 55 papers were chosen as the primary studies for our SLR, as shown in Table 4.

3.4 Quality assessment

To assess the overall quality of the studies included in this review, as well as the strength of the evidence they contribute, we developed a set of quality evaluation questions. These questions were designed to critically evaluate the methodological rigor and reporting transparency of each study. The questions we used to guide this process are as follows:

- Q1: Is the research process clearly documented, allowing for transparency and reproducibility?
- Q2: Are the metrics and dimensions used to assess scalability both valid and reliable?
- Q3: Does the study adequately address all of the research questions posed?
- Q4: Are the research objectives explicitly stated and aligned with the study's methodology?
- Q5: Does the study adequately discuss its limitations and suggest directions for future research?

Each study's response to these questions was categorized as 'yes,' 'to some extent,' or 'no.' We assigned numerical scores based on these categories: 1 for 'yes,' 0.5 for 'to some extent,' and 0 for 'no.' The total quality score for each study was then calculated by summing the individual scores for each question. These scores were used to guide the synthesis of findings in the later stages of the review.

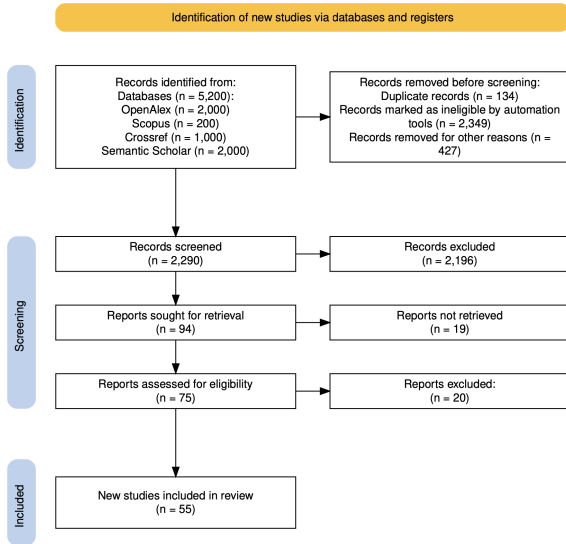


Figure 1: PRSIMA Flow diagram

3.5 Data extraction and synthesis

Data was systematically extracted from the selected studies, including key details such as research type, publication year, and statistical data for subsequent analysis. Additionally, we extracted information regarding the various scaling approaches, techniques, and metrics employed for autoscaling in microservices architectures.

4 Results

Figure 2 illustrates the distribution of the selected papers according to their publication type. The majority of papers (63.63%) were published as conference and proceedings papers, totaling 35 papers. This was followed by journal articles, with 19 papers (34.55%), and a single preprint paper, accounting for 1.82%.

Further analysis of the selected papers by publication year, as shown in Figure 3, reveals a marked increase in the number of publications in recent years. This trend highlights the growing interest and evolving focus in this research domain.

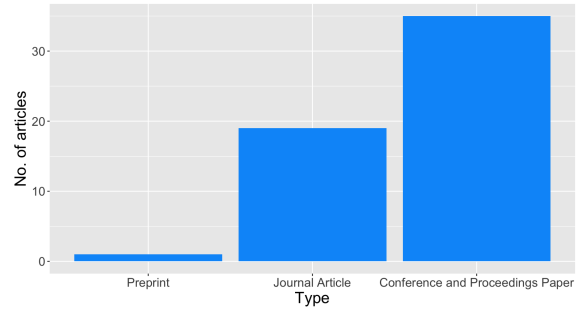


Figure 2: Distribution of articles based on article type

4.1 RQ1: What types of research have been conducted in the scalability domain of microservices?

To address RQ1, we categorized the selected studies as summarized in Table 3. This classification highlights the distribution of research papers across various domains, including load testing, orchestration, decomposition and discovery, benchmarking, development and deployment, load balancing, and autoscaling. While categorizing, we ensured that each paper was included in a single category based on its primary focus or area of emphasis. For instance, if a paper discussed both load balancing and autoscaling, we classified it under the category where it had a stronger emphasis, ensuring clarity and avoiding overlap in the classification.

The findings show that while scalability in microservices has been explored from several perspectives, autoscaling techniques constitute a significant proportion of the research. However, other areas have also garnered attention, reflecting the diversity of challenges in achieving scalable microservices architectures.

1. Load Testing:

A key area of research is load testing, with studies focusing on tools and techniques for assessing the scalability of microservices under different workloads. For instance, [24] introduced Scalar, a tool that addresses limitations in existing load testing frameworks, such as scalability analysis and inter-machine communication. Scalar's features, such as real-time monitoring and visualization of results, make it useful for evaluating performance under high loads. Load testing is crucial for understanding system behavior and identifying bottlenecks during scaling.

2. Orchestration:

In the domain of orchestration, studies have explored frameworks and platforms that manage microservices interactions to enhance scalability. For example, [25] introduced Beethoven, a platform designed to simplify the orchestration of microservices without prior registration. Beethoven dynamically composes new microservices, improving scalability when deploying large systems.

Table 2: Inclusion and Exclusion Criteria

Inclusion Criteria (IC)	
IC1	Studies published in English.
IC2	Studies that assess the scalability of microservices architecture.
Exclusion Criteria (EC)	
EC1	Studies not published in English
EC2	Studies that mentions microservices but does not specifically address scalability
EC3	Documents which are theses, books and keynote talk abstracts

Similarly, [26] proposed COCOS, a containerized architecture designed to support efficient and distributed control of heterogeneous systems, ensuring reliable resource scalability. Moreover, [27] proposed DOCMA, a decentralized microservices orchestrator using peer-to-peer principles for scalability, resilience, and self-healing in edge computing.

3. Decomposition and Discovery:

The decomposition and discovery category focuses on strategies for designing and organizing scalable microservices architectures. [28] proposed a scalability pattern language, organizing patterns according to different axes, such as layers, components, and shards. This approach provides a structured vocabulary for scalability decisions in microservices systems. [29] also explored service decomposition, recommending the splitting of applications into smaller, granular microservices that are loosely coupled to ensure better scalability and resilience.

4. Benchmarking:

Benchmarking studies aim to assess the scalability of microservices and related systems. [30] introduced μ qSim, a queueing network simulator, designed to model and evaluate scalability factors such as throughput and latency for microservices. This tool allows researchers to model complex interactions between microservices and identify performance bottlenecks. Similarly, [31] proposed a benchmarking method tailored for cloud-native applications, providing metrics and techniques to empirically assess scalability under various cloud deployment scenarios.

5. Development and Deployment:

Several studies emphasize tools and practices for ensuring scalable microservices deployments. [32] examined the deployment strategies used at otto.de, highlighting the importance of containerization technologies like Apache Mesos to enhance scalability and reliability. [33] discussed the use of dedicated programming languages, such as Jolie, to simplify microservices development and support non-uniform scalability across services. Other studies, such as [34] and [35], proposed frameworks for real-time performance monitoring and resource demand esti-

mation, helping organizations optimize their deployment strategies and resource allocation.

6. Load Balancing:

The load balancing category has been actively researched, with several studies proposing algorithms and techniques to optimize resource allocation and response times. [36] introduced the Scalability Improvement System (SIS), a load balancing system that dynamically distributes workload between caches to reduce delays and improve system performance. Other studies, such as [37] and [38], proposed decentralized and chain-oriented load balancing algorithms designed to optimize request routing in microservices architectures. Additionally, [39] introduced TCLBM, an algorithm that balances load based on task chains and system resource usage, addressing interdependencies among microservices.

7. Autoscaling:

Finally, autoscaling remains a major area of focus. The studies in this category were analyzed based on the type of scaling (reactive, proactive, or hybrid), scaling policies, metrics considered, techniques employed, and virtualization methods. These findings are summarized in Table 5, which provides a detailed comparison of the various approaches to autoscaling. There is a growing trend towards proactive autoscaling methods, with machine learning and deep learning being leveraged for predictive scaling. This shift from traditional reactive approaches towards data-driven, adaptive techniques marks an important development in autoscaling research. Many studies explored custom metrics, such as message queue length and latency, which were considered more effective than conventional resource utilization metrics in predicting scaling requirements. Furthermore, several studies, such as [40, 41, 35], addressed the challenges of scaling in hybrid cloud environments, providing a flexible and cost-effective solution for large-scale systems.

In summary, the research on microservices scalability has evolved to address a wide range of areas. While autoscaling remains the most prominent area of focus, there is substantial research in other areas, such as load testing, orchestration, decomposition, benchmarking, and load balancing. The recent trend in autoscaling

focuses particularly on proactive approaches that incorporate advanced techniques like machine learning and deep learning, signaling a move towards more intelligent, adaptive systems that anticipate resource needs.

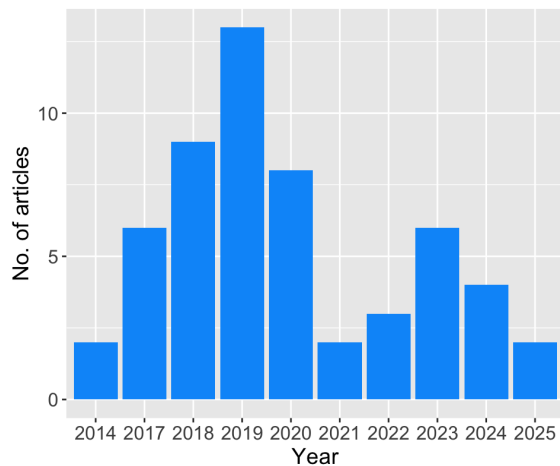


Figure 3: Distribution of articles based on year

4.2 RQ2: What are the major challenges faced by researchers while trying to scale microservices architectures?

To address RQ2, we identified key challenges mentioned by various authors in the reviewed studies. Several studies have explored the difficulties associated with scaling microservices architectures, highlighting the following major challenges:

1. Difficulty in service metrics collection:

As mentioned by [42, 43], a significant challenge in scaling microservices is the difficulty in collecting service metrics. Many cloud platforms today lack the necessary tools or granularity for monitoring service performance and resource usage at a microservices level.

2. Challenge to optimize both performance and cost:

[42, 43] highlighted that efficiently scaling microservices without over-provisioning (which wastes resources and increases costs) or under-provisioning (leading to service degradation or failure) remains a major challenge. [44] explored threshold-based autoscaling using message queues but identified difficulties such as flooded queues. They also noted challenges in leveraging message queue state information to prevent both under- and over-provisioning of microservices, which can cause service delays that are difficult to recover from.

Few studies, such as SCAD [35] and TERA-Scaler [45], employed multi-objective optimiza-

tion to balance cost and performance in microservices auto-scaling. SCAD formulated resource allocation as a multi-objective problem, optimizing API performance, availability, and cloud hosting costs using genetic algorithms. Similarly, TERA-Scaler adopted a multi-criteria strategy, proactively scaling microservices based on CPU usage, memory, and input data rates to reduce costs while maximizing system performance. These works highlighted that while optimization strategies existed, balancing performance and cost remained a complex challenge, requiring trade-offs between system efficiency and economic constraints.

3. Difficulty to determine the need for scaling:

As discussed by [42, 43], the large number of interconnected microservices makes it difficult to accurately determine when scaling is required. This dynamic environment often leads to challenges in predicting resource demands and scaling thresholds. [46] observed that delays in container startup impact auto-scaling efficiency, making timely scaling difficult. In the same vein, [47] highlighted that incorrect scaling predictions lead to performance issues and resource wastage, proposing a proactive-reactive approach to improve accuracy.

4. Security:

As microservices scale, it is crucial that their security mechanisms, including authorization and authentication, scale as well while maintaining high levels of robustness and availability. [40] addressed this challenge by proposing an authorization pattern that balances both scalability and security objectives, ensuring that security measures do not become a bottleneck as the system grows.

5. Need for customized scaling approaches:

As highlighted by [48], the varying Quality of Service (QoS) requirements across different microservices make it impossible to apply a one-size-fits-all autoscaling approach. This necessitates the development of customized scaling strategies tailored to the specific needs of each service, complicating scalability efforts.

6. Risk of cascading QoS violations:

According to [49], the stricter tail latency requirements for individual microservices increase the risk of cascading QoS violations. Due to the complex interdependencies between microservices, a single misbehaving service can trigger cascading delays and failures, severely impacting overall system performance. Furthermore, [47] described the domino effect, where traditional local scaling approaches may unintentionally trigger scaling

actions across multiple services, leading to inefficiencies, delays in adapting to workload changes, and increased resource consumption.

7. **Fluctuating workload:**

Fluctuating workloads, as discussed in studies [50, 51, 52], introduce significant challenges in managing response times and adhering to Service Level Agreements (SLAs). Rapid changes in request rates can make it difficult to predict when to scale, complicating efforts to ensure both cost-effectiveness and optimal performance, as further highlighted by [53]. Additionally, [54] highlighted the challenge of managing sudden bursts in real-time, proposing a burst-aware autoscaler that uses forecasting and prediction to reduce SLO violations. Furthermore, [55] noted that fluctuating workloads can cause longer response times and service unavailability, highlighting limitations in current autoscalers, such as fixed resource limits, slow response to changes, and architectural vulnerabilities.

8. **Challenge of identifying the right scaling parameters under a dynamic environment:**

As noted by [56], autoscaling in dynamic environments poses challenges in determining the right scaling parameters, including the need for sophisticated modeling expertise to align scaling decisions with SLAs. Their study on real-time cloud applications highlighted the importance of prioritizing response time as the primary QoS metric and proposed a versatile autoscaling model to predict peak resource demands and optimize microservices scaling. Research referenced in [47] highlighted that the dynamic, distributed nature of microservices challenged efficient resource scheduling, which is crucial for maintaining service quality under varying conditions.

9. **Monitoring:**

According to [32], the growing number of microservices introduces significant challenges in monitoring and anomaly detection. To tackle these challenges, the authors described Otto.de's organizational approach, which uses vertical decomposition and cross-functional feature teams to enhance scalability and development capacity. Similarly, [34] highlighted the difficulty of monitoring microservices architectures, emphasizing that the complex interdependencies between services make quality assurance even more demanding. Additionally, [57] highlighted that existing monitoring systems often struggle to support real-time auto-scaling, and the large volumes of data generated by these tools increase the complexity of managing metrics effectively.

In summary, the major challenges in achieving scalable microservices architectures revolve around the com-

plexities of service metrics collection, dynamic scaling decision-making, and maintaining performance while managing cost and security.

Additionally, fluctuating workloads, the risk of cascading QoS violations, and the need for customized scaling approaches add further layers of difficulty, highlighting the importance of sophisticated autoscaling models and effective monitoring in real-world microservices environments.

4.3 **RQ3: What are the most commonly used tools in microservices architectures?**

To address RQ3, we identified various tools used in microservices architecture, which support deployment, orchestration, monitoring, autoscaling, and load balancing. Below, we categorize and describe the most commonly used tools identified in our studies:

1. **Containerization Tools:**

From the primary studies, we identified several containerization tools that enable lightweight, portable, and scalable deployment of microservices. Docker, OpenVZ, and Linux Containers (LXC) were the tools mentioned across the studies. Additionally, cloud-based containerization services, including Elastic Container Service (ECS), Google Kubernetes Engine (GKE), and Azure Container Services, were also referenced, which facilitate scalable container deployment in cloud environments.

2. **Load Balancers:**

Load balancing in microservices efficiently distributes traffic across services to prevent overload and ensure high availability. From the primary studies, we identified various software-based load balancers, such as Nginx, Apache HTTP Server, Envoy, and Ribbon. Managed cloud-based services like Ingress and Elastic Load Balancer (ELB) were also mentioned. A few research-based solutions, such as COLBA [38] and TCLBM [39], were highlighted.

3. **Load Testing Tools:**

Load testing tools simulate heavy traffic to assess microservices performance and scalability. Primary studies highlighted JMeter, Gatling, Grinder, Locust, and k6. Custom tools, such as Scalar [24], a distributed load generator, were also noted. Additional tools like httpperf, ApacheBench, and Terminus were used for measuring response times and throughput. Chaos engineering tools like Netflix's Chaos Monkey were used to test resilience.

Table 3: Summary of Study Categories

Category	Total	Percentage	Studies
Load Testing	1	1.82%	[24]
Orchestration	3	5.45%	[25, 26, 27]
Decomposition and Discovery	2	3.64%	[28, 29]
Benchmarking	2	3.64%	[30, 31]
Development and Deployment	4	7.27%	[32, 33, 34, 35]
Load Balancing	5	9.09%	[36, 37, 58, 38, 39]
Auto Scaling	38	69.09%	[59, 40, 60, 61, 54, 62, 63, 64, 46, 57, 47, 55, 53, 44, 50, 41, 65, 66, 42, 67, 68, 48, 69, 70, 49, 51, 71, 72, 43, 56, 73, 74, 75, 76, 45, 52, 77, 78]

Table 4: Selected studies (sorted based on publication year)

ID	Ref.	Year	Title
S1	[36]	2014	Evaluating PaaS scalability and improving performance using scalability improvement systems.
S2	[24]	2014	Scalar: Systematic scalability analysis with the universal scalability law.
S3	[37]	2017	A decentralized system for load balancing of containerized microservices in the cloud.
S4	[32]	2017	Microservice architectures for scalability, agility, and reliability in e-commerce.
S5	[59]	2017	Elascale: Autoscaling and monitoring as a service.
S6	[58]	2017	A scalable routing mechanism for stateful microservices.
S7	[40]	2017	Highly scalable microservice-based enterprise architecture for smart ecosystems in hybrid cloud environments.
S8	[60]	2017	Auto-scaling of containers: The impact of Relative and Absolute metrics.
S9	[38]	2018	Load balancing across microservices.
S10	[53]	2018	Auto-scaling microservices on IaaS under SLA with cost-effective framework.
S11	[44]	2018	Investigating performance metrics for scaling microservices in cloud-IoT environments.
S12	[50]	2018	CAUS: An elasticity controller for a containerized microservice.
S13	[33]	2018	Microservices: How to make your application scale.
S14	[25]	2018	Beethoven: An event-driven lightweight platform for microservice orchestration.
S15	[28]	2018	A pattern language for scalable microservices-based systems.
S16	[41]	2018	Monitoring-based auto-scalability across hybrid clouds.
S17	[65]	2018	Cloud software performance metrics collection and aggregation for auto-scaling module.
S18	[66]	2019	Performance modeling for cloud microservice applications.
S19	[42]	2019	Microscaler: Automatic scaling for microservices with an online learning approach.
S20	[67]	2019	A study on performance measures for auto-scaling CPU-intensive containerized applications.
S21	[68]	2019	Hyscale: Hybrid and network scaling of dockerized microservices in cloud data centres.
S22	[48]	2019	Agnostic approach for microservices autoscaling in cloud applications.
S23	[30]	2019	μQsim: Enabling accurate and scalable simulation for interactive microservices.
S24	[69]	2019	Multilayered autoscaling performance evaluation: Can virtual machines and containers co-scale?
S25	[70]	2019	Capacity-driven scaling schedules derivation for coordinated elasticity of containers and virtual machines.
S26	[29]	2019	Design of scalable and resilient applications using microservice architecture in PaaS cloud.
S27	[49]	2019	Leveraging deep learning to improve the performance predictability of cloud microservices.
S28	[43]	2019	Microscaler: Cost-effective scaling for microservice applications in the cloud with an online learning approach.
S29	[61]	2019	Cost aware resource sizing and scaling of microservices.
S30	[27]	2019	DOCMA: A decentralized orchestrator for containerized microservice applications.
S31	[54]	2020	Burst-aware predictive autoscaling for containerized microservices.
S32	[62]	2020	Towards a client-centric QoS auto-scaling system.
S33	[71]	2020	Adaptive microservice scaling for elastic applications.
S34	[72]	2020	A quantitative approach for estimating the scaling thresholds and step policies in a distributed microservice architecture.
S35	[26]	2020	Cocos: A scalable architecture for containerized heterogeneous systems.
S36	[34]	2020	Research on microservice application performance monitoring framework and elastic scaling mode.
S37	[51]	2020	Predictive autoscaling of microservices hosted in fog microdata center.
S38	[39]	2020	TCLBM: A task chain-based load balancing algorithm for microservices.
S39	[56]	2021	Intelligent autoscaling of microservices in the cloud for real-time applications.
S40	[73]	2021	PHPA: A proactive autoscaling framework for microservice chain.
S41	[31]	2022	A configurable method for benchmarking scalability of cloud-native applications.
S42	[74]	2022	Graph-phpa: Graph-based proactive horizontal pod autoscaling for microservices using LSTM-GNN.
S43	[75]	2022	Improved Q network auto-scaling in microservice architecture.
S44	[76]	2023	An auto-scaling approach for microservices in cloud computing environments.
S45	[45]	2023	Tera-scaler for a proactive auto-scaling of e-business microservices.
S46	[35]	2023	Scad: Scalability advisor for interactive microservices on hybrid clouds.
S47	[52]	2023	Proactive random-forest autoscaler for microservice resource allocation.
S48	[77]	2023	Trace-driven scaling of microservice applications.
S49	[78]	2023	Gym-HPA: Efficient auto-scaling via reinforcement learning for complex microservice-based applications in Kubernetes.
S50	[63]	2024	Enhancing performance and scalability in microservices.
S51	[64]	2024	MSARS: A meta-learning and reinforcement learning framework for SLO resource allocation and adaptive scaling for microservices.
S52	[46]	2024	Smart HPA: A resource-efficient horizontal pod auto-scaler for microservice architectures.
S53	[57]	2024	Proactive auto-scaling for service function chains in cloud computing based on deep learning.
S54	[47]	2025	Proactive-reactive microservice architecture global scaling.
S55	[55]	2025	Towards resource-efficient reactive and proactive auto-scaling for microservice architectures.

Table 5: Comparison of Papers in Auto Scaling Domain

Serial No.	Article	Type	Policy	Metrics	Technique	Approach	Virtualization
1	[59]	Reactive	Horizontal	CPU ¹ usage, Memory usage and Network utilization	Threshold based	QoS ² aware	VM ³ + Container
2	[40]	Reactive	Horizontal	CPU usage, Memory utilization and Custom metrics including internal queue length, progress of an algorithm etc.	Threshold based	Cost aware	VM + Container
3	[53]	Proactive	Horizontal	CPU usage and Memory utilization	ML ⁴	Cost and SLA ⁵ aware	Container
4	[44]	Reactive	Horizontal	CPU utilization and Message Queue	Threshold based	SLA Aware	Container
5	[50]	Hybrid ⁶	Horizontal	Load intensity and Request queue	Threshold based	Workload	Container
6	[41]	Reactive	Horizontal	CPU usage, Memory usage, Network usage, Disk usage, and SLA	Threshold based, Time series analysis and Queuing theory	Budget aware and SLA aware	Container
7	[65]	Reactive	Horizontal	QoS Metrics (Infrastructure and Application Levels): Response times, Successful/Failed requests	Threshold based	SLA aware	Container
8	[66]	Hybrid	Horizontal	CPU usage, Response time and Request success rate	Threshold based and Regression model	SLO ⁷ aware	Container
9	[42]	Reactive	Horizontal	QoS metrics: service invocation and service latency	Bayesian Optimization and online learning with a heuristic method	SLA aware and Cost aware	Container
10	[67]	Reactive	Horizontal	Absolute and Relative CPU utilization	Threshold based	QoS aware	Container
11	[68]	Reactive	Hybrid ⁸	CPU and Memory usage	Threshold based	SLA aware	Container
12	[48]	Reactive	Horizontal	CPU utilization, Memory usage and Number of undelivered messages in a queue	Threshold based	QoS aware	Container
13	[69]	Reactive	Hybrid	CPU usage	Threshold based	QoS Aware	VM + Container
14	[70]	Proactive	Hybrid	Forecasted request rate, Microservice capacity, Resource utilization	-	Cost aware	VM + Container
15	[49]	Proactive	Horizontal	Throughput and Tail latency	ML (DL ⁹)	QoS Aware	Container
16	[51]	Proactive	Horizontal	CPU utilization, number of requests and response time	ML	Resource and SLA aware	Container
17	[71]	Proactive	Horizontal	Response time	ML and Time series analysis	Cost aware	Container
18	[72]	Reactive	Horizontal	CPU utilization	Threshold based	Load aware	Container
19	[43]	Reactive	Horizontal	QoS	Heuristic approach and online learning approach	Cost aware and SLA aware	Container
20	[56]	Proactive	Horizontal	CPU usage, Memory usage, Queue and Response time	ML (RL ¹⁰)	Resource aware	Container
21	[73]	Proactive	Horizontal	Latency and CPU usage	ML (GNN ¹¹)	SLO aware, Cost aware, QoS aware	Container

Continued on next page

- ¹Central Processing Unit
- ²Quality of Service
- ³Virtual Machine
- ⁴Machine Learning
- ⁵Service Level Agreement
- ⁶Reactive + Proactive
- ⁷Service Level Objective
- ⁸Vertical + Horizontal
- ⁹Deep Learning
- ¹⁰Reinforcement Learning
- ¹¹Graph Neural Network

Table 5 – continued from previous page

Serial No.	Article	Type	Policy	Metrics	Technique	Approach	Virtualization
22	[74]	Proactive	Horizontal	vCPU usage, Memory usage and Workload	ML (GCN ¹² and LSTM ¹³)	-	Container
23	[75]	Reactive	Horizontal	No. of requests	ML (AI ¹⁴ and RL)	Resource aware	Container
24	[76]	Hybrid	Horizontal	Resource utilization (CPU and Memory) and Workload patterns	ML and Threshold based	Workload aware	Container
25	[45]	Proactive	Hybrid	CPU usage, Memory usage, Input data rate	Dependency based	Cost aware and Performance aware	Container
26	[52]	Proactive	Hybrid	CPU and Memory utilization	ML (Random Forest)	QoS aware and SLA aware	Container
27	[77]	Reactive	Hybrid	CPU and Memory utilization	Threshold based	SLA aware	Container
28	[78]	Hybrid	Horizontal	Response time, CPU and Memory utilization	ML and Threshold based	-	Container
29	[60]	Reactive	Horizontal	Absolute and relative metrics	Threshold based	-	Container
30	[61]	Proactive	Horizontal	Workload demand, resource costs and application performance	Heuristic algorithm	Cost aware	Container
31	[54]	Hybrid	Horizontal	Processed requests, SLO violations, scale operations and cost	ML (CART ¹⁵ and EN ¹⁶)	Burst aware	Container
32	[62]	Reactive	Horizontal	Compute relative metrics, infrastructure state information and network related metrics	Threshold based	QoS aware	Container
33	[63]	Proactive	Horizontal	CPU utilization, network utilization, workload intensity and end-to-end tail latency	ML (QGPR ¹⁷)	Workload and SLO aware	Container
34	[64]	Proactive	Hybrid	Workload status, microservice chain structure, response times, instance sizes, CPU utilization and load-latency	ML (Meta-learning and RL)	QoS and SLO aware	Container
35	[46]	Reactive	Horizontal	CPU usage and response time	Threshold based	Workload aware	Container
36	[57]	Proactive	-	CPU, memory and bandwidth usage for VNFs, ¹⁸	DL (MLP ¹⁹ and LSTM)	Workload aware	Container
37	[47]	Hybrid	Horizontal	MCL ₂₀	Threshold based and MLP	Workload aware	Container
38	[55]	Hybrid	Horizontal	Resource utilization, current replica count and SLA metrics	Threshold based and ML	Resource aware	Container

- ¹²Graph Convolution Network
- ¹³Long Short Term Memory
- ¹⁴Artificial Intelligence
- ¹⁵Classification and Regression Trees
- ¹⁶Elastic Net Regression
- ¹⁷Quantum Gaussian Process Regression
- ¹⁸Virtual Network Functions
- ¹⁹Multilayer Perceptron
- ²⁰Maximum Computational Load

4. **Orchestration Tools:**

Orchestration tools automate the deployment, scaling, and management of containerized microservices. Key tools identified include Kubernetes, Apache Mesos, Docker Swarm, Marathon, and OpenShift. Managed services like Azure Kubernetes Service (AKS), GKE, and Amazon Elastic Kubernetes Service (EKS) simplify cluster management. Some studies also proposed custom solutions, such as Beethoven [25], CO-COS [26], and DOCMA [27].

5. **Autoscalers:**

Autoscalers dynamically adjust computing resources based on real-time demand to optimize performance and cost. From our primary studies, we found KHPA to be the most widely used autoscaler. In addition, several custom autoscalers were proposed across the studies, including Elascalle [59], CAUS [50], Microscaler [42, 43], KHPA-A [67], TRIM [77], pHPA [73], GraphHPA [74], TERA-Scaler [45], SCAD [35], gym-hpa [78], RF proactive autoscaler [52], MSARS [64], SmartHPA [46], and ProSmartHPA [55].

6. **Metrics Collection, Monitoring and Visualization:**

Monitoring and metrics collection are vital for assessing the health and performance of microservices. We identified tools like Prometheus and cAdvisor, which provide real-time insights into containerized workloads. For visualization, Grafana was commonly used to create monitoring dashboards that help operators analyze system performance. Additionally, the Elastic Stack (Elasticsearch, Logstash, and Kibana – ELK) was frequently employed for log analysis and centralized monitoring in microservices architectures.

7. **Service Mesh and Service Discovery:**

Service mesh solutions facilitate secure, reliable, and observable communication between microservices. Istio and Consul were the most commonly identified tools for implementing service meshes. Additionally, we found that Consul, Netflix Eureka, and Apache Zookeeper were used as service discovery tools.

8. **Open-source Benchmarking Applications:**

To assess the scalability of microservices-based architectures, researchers utilized various open-source benchmarking tools, including Robot Shop²¹, DeathStarBench²², Google's Online Boutique²³, Hipster Shop²⁴, SockShop²⁵, Is-

tio's Bookinfo²⁶, Go-microservices²⁷, TeaStore²⁸, Movie App²⁹, and Pwitter³⁰, as used in the primary studies. Additionally, [31] introduced Theodolite³¹, a benchmarking tool designed for conducting empirical scalability evaluations of cloud-native applications, frameworks, and deployment strategies.

This section explores various tools and technologies integral to microservices architectures. The most commonly used tools identified include Docker for containerization, Kubernetes for orchestration, KHPA for autoscaling, Locust for load testing, Istio for service mesh, and Prometheus for metrics collection and monitoring.

5 Discussion

This SLR has provided important insights into the types of research conducted on the scalability of microservices, the most commonly used tools and technologies in microservices architecture, and the challenges researchers face in this area. Various strategies have been explored, including autoscaling mechanisms that adjust resources based on demand, container orchestration for efficient management of applications, and design patterns that support modularity. Our analysis reveals that a diverse range of tools and technologies are essential for supporting scalability in microservices architectures. These tools span across areas such as containerization, orchestration, service discovery, service mesh, benchmarking, load testing, monitoring, logging, and metrics collection. While many studies rely on well-established open-source solutions, there is a notable shift toward the development of custom autoscalers and orchestration strategies. We found that most custom autoscalers leverage predictive approaches, often combining both reactive and proactive methods. This trend highlights the growing demand for more sophisticated tools that can intelligently adapt to the dynamic needs of scalable microservices environments. Furthermore, the review highlights challenges such as the complexities of managing distributed systems and ensuring data consistency. A growing trend is the use of machine learning techniques for predictive scaling, which aims to improve resource allocation in response to changing workloads.

Machine learning models, particularly those utilizing time-series analysis, reinforcement learning, and deep learning techniques, are increasingly being adopted to predict and optimize scaling require-

²¹<https://github.com/instana/robot-shop>

²²<https://github.com/delimitrou/DeathStarBench>

²³<https://github.com/GoogleCloudPlatform/microservices-demo>

²⁴<https://github.com/lightstep/hipster-shop>

²⁵<https://github.com/microservices-demo/microservices-demo>

²⁶<https://istio.io/latest/docs/examples/bookinfo/>

²⁷<https://github.com/harlow/go-micro-services>

²⁸<https://github.com/DescartesResearch/TeaStore>

²⁹<https://github.com/Chrisztian/cinema-microservice/tree/master/movies-service>

³⁰<https://github.com/deib-polimi/Pwitter>

³¹<https://github.com/cau-se/theodolite>

ments in environments with fluctuating demand patterns. This shift from traditional reactive approaches to data-driven, adaptive methodologies represents a significant advancement in autoscaling research, enabling more precise and dynamic resource management.

However, we have identified a significant gap in the lack of standardized guidelines for selecting the most effective performance-related and cost-related metrics for autoscaling. This gap arises partly because microservices environments are highly heterogeneous, and the effectiveness of different metrics can vary based on the specific use case, workloads, and system architectures. The absence of universal guidelines makes it difficult to compare autoscaling strategies across studies and hinders the development of universally applicable scaling solutions. Furthermore, we observed that there is no established method to address situations where systems provide inaccurate scaling metrics or factors. In real-world cloud environments, issues such as network latency, node failures, and resource contention can cause significant deviations in reported scaling metrics, leading to suboptimal scaling decisions. Therefore, there is an urgent need for autoscaling mechanisms that are not only robust to metric inconsistencies but also capable of adapting to the unpredictable nature of real-world systems.

This review is not without its limitations. The search strategy was confined to specific databases and utilized a limited set of search terms, which may have restricted the breadth of the literature reviewed. Future reviews could benefit from a broader search across additional databases, incorporating a wider range of keywords and including grey literature to capture emerging trends that may not be covered in traditional peer-reviewed sources.

The implications of this review are significant for both researchers and practitioners in the field of software engineering. For researchers, the findings underscore the importance of developing comprehensive frameworks that guide the implementation of scalable microservices. Future research should focus on creating standardized performance and cost metrics that can be applied across different microservices architectures. This will foster greater consistency in benchmarking and improve the comparability of scalability solutions. For practitioners, understanding the current trends and challenges in microservices scalability is essential for informed decision-making. Practitioners should consider adopting a hybrid scaling approach, combining proactive and reactive methods, and experiment with custom scaling metrics that are more closely aligned with their unique workload patterns. Organizations must also invest in tools and frameworks that can handle the inherent complexities of distributed microservices systems.

In conclusion, while significant progress has been made in addressing the scalability challenges of microservices, many issues remain unresolved, particu-

larly around the standardization of autoscaling metrics and handling inaccuracies in scaling data. Future research should focus on creating robust, adaptive autoscaling systems that can handle real-world complexities and dynamic workloads. Additionally, a more holistic approach to scalability that considers both performance and cost optimization, along with system resilience, will be crucial for the next generation of microservices architectures.

6 Conclusions and future work

In conclusion, our systematic literature review presents a comprehensive overview of the current state of research on scalability in the microservices domain. Initially, we considered 5,200 articles from various databases, which were narrowed down to 2,290 after removing duplicates and ineligible records. After applying our inclusion and exclusion criteria and conducting a thorough screening process, we ultimately selected 55 scholarly articles for rigorous analysis. Through this analysis, we have identified key strategies, tools, technologies, emerging trends, and challenges associated with scaling microservices architectures.

Our findings reveal a diverse body of literature addressing scalability in microservices, with a significant focus on autoscaling mechanisms. Proactive approaches using machine learning are becoming increasingly prominent, reflecting a shift toward predictive and adaptive scaling strategies. However, achieving scalable microservices architectures remains challenging. Key difficulties include collecting accurate metrics, making dynamic scaling decisions, balancing performance with cost and security, managing workload variability, and mitigating risks of cascading QoS violation. We also identified the most commonly used tools in microservices architectures, with Docker, Kubernetes, KHPA, Locust, Istio, and Prometheus being the most popular. Additionally, we included the benchmarking applications referenced in prior research for experimenting with microservices.

In our future work, we plan to develop an autoscaling approach that accounts for delayed, inaccurate, or incomplete metrics while considering inter-service dependencies. We aim to validate the effectiveness of this approach in real-world scenarios. By focusing on these areas, we hope to make meaningful contributions to the scalability of microservices architectures.

Authors' contribution

The authors confirm contribution to the paper as follows: NG: Conceptualization, Methodology, Data Curation, Formal Analysis, Writing – Original Draft, Visualization, Software; SS: Supervision, Writing – Review & Editing, Validation, Project Administration; RC: Supervision, Writing – Review & Editing, Validation and Project Administration. All authors reviewed the results and approved the final version of

the manuscript.

Competing interests

The authors have declared that no competing interests exist.

References

- [1] V. L. Nogueira, F. S. Felizardo, A. M. Amaral, W. K. Assuncao, and T. E. Colanzi, "Insights on microservice architecture through the eyes of industry practitioners," *arXiv preprint arXiv:2408.10434*, 2024. [Online]. Available: <https://doi.org/10.48550/arXiv.2408.10434>
- [2] G. Blinowski, A. Ojdowska, and A. Przybyłek, "Monolithic vs. microservice architecture: A performance and scalability evaluation," *IEEE Access*, vol. 10, pp. 20 357–20 374, 2022. [Online]. Available: <https://doi.org/10.1109/ACCESS.2022.3152803>
- [3] N. Bjørndal, A. Bucchiarone, M. Mazzara, N. Dragoni, S. Dustdar, F. B. Kessler, and T. Wien, "Migration from monolith to microservices: Benchmarking a case study," Tech. Rep., 2020.[Online]. Available: <https://www.researchgate.net/profile/...>, Tech. Rep., 2020. [Online]. Available: <https://doi.org/10.13140/RG.2.2.27715.14883>
- [4] R. Capuano and H. Muccini, "A systematic literature review on migration to microservices: a quality attributes perspective," in *2022 IEEE 19th International Conference on Software Architecture Companion (ICSA-C)*. IEEE, 2022, pp. 120–123. [Online]. Available: <https://doi.org/10.1109/ICSA-C54293.2022.00030>
- [5] S. Hussein, M. Lahami, and M. Torjmen, "Assessing the quality of microservice and monolithic-based architectures: A systematic literature review," *Operational Research in Engineering Sciences: Theory and Applications*, vol. 7, no. 2, 2024. [Online]. Available: <https://doi.org/0.31181/oresta/070220>
- [6] A. Hilali, H. Hafiddi, and Z. El Akkaoui, "Microservices adaptation using machine learning: A systematic mapping study," *ICSOF*, pp. 521–532, 2021.
- [7] S. Hassan, R. Bahsoon, and R. Kazman, "Microservice transition and its granularity problem: A systematic mapping study," *Software: Practice and Experience*, vol. 50, no. 9, pp. 1651–1681, 2020. [Online]. Available: <https://doi.org/10.1002/spe.2869>
- [8] M. S. Hamzehloui, S. Sahibuddin, and K. Salah, "A systematic mapping study on microservices," in *Recent Trends in Data Science and Soft Computing: Proceedings of the 3rd International Conference of Reliable Information and Communication Technology (IRICT 2018)*. Springer, 2019, pp. 1079–1090. [Online]. Available: https://doi.org/10.1007/978-3-319-99007-1_100
- [9] Z. Stojanov, I. Hristoski, J. Stojanov, and A. Stojkov, "Research trends and recommendations for future microservices research," , vol. 36, no. 1, pp. 105–130, 2024. [Online]. Available: [https://doi.org/10.15514/ISPRAS-2024-36\(1\)-7](https://doi.org/10.15514/ISPRAS-2024-36(1)-7)
- [10] S. Alharthi, A. Alshamsi, A. Alseiyari, and A. Alwarafy, "Auto-scaling techniques in cloud computing: Issues and research directions," *Sensors*, vol. 24, no. 17, p. 5551, 2024. [Online]. Available: <https://doi.org/10.3390/s24175551>
- [11] I. Ghani, W. M. Wan-Kadir, A. Mustafa, and M. I. Babir, "Microservice testing approaches: A systematic literature review," *International Journal of Integrated Engineering*, vol. 11, no. 8, pp. 65–80, 2019. [Online]. Available: <https://doi.org/10.30880/ijie.2019.11.08.008>
- [12] I. K. Aksakalli, T. Çelik, A. B. Can, and B. Tekinerdoğan, "Deployment and communication patterns in microservice architectures: A systematic literature review," *Journal of Systems and Software*, vol. 180, p. 111014, 2021. [Online]. Available: <https://doi.org/10.1016/j.jss.2021.111014>
- [13] P. Haindl, P. Kochberger, and M. Svegggen, "A systematic literature review of inter-service security threats and mitigation strategies in microservice architectures," *IEEE Access*, 2024. [Online]. Available: <https://doi.org/10.1109/ACCESS.2024.3406500>
- [14] D. Taibi, V. Lenarduzzi, and C. Pahl, "Architectural patterns for microservices: a systematic mapping study," in *CLOSER 2018: Proceedings of the 8th International Conference on Cloud Computing and Services Science; Funchal, Madeira, Portugal, 19-21 March 2018*. SciTePress, 2018. [Online]. Available: <https://doi.org/19/03/201821/03/2018>
- [15] —, "Continuous architecting with microservices and devops: A systematic mapping study," in *Cloud Computing and Services Science: 8th International Conference, CLOSER 2018, Funchal, Madeira, Portugal, March 19-21, 2018, Revised Selected Papers 8*. Springer, 2019, pp. 126–151. [Online]. Available: https://doi.org/10.1007/978-3-030-29193-8_7
- [16] M. Söylemez, B. Tekinerdogan, and A. Kolukisa Tarhan, "Challenges and solution directions of microservice architectures: A systematic literature review," *Applied sciences*, vol. 12, no. 11, p. 5507, 2022. [Online]. Available: <https://doi.org/10.3390/app12115507>
- [17] H. Vural, M. Koyuncu, and S. Guney, "A systematic literature review on microservices," in *Computational Science and Its Applications—ICCSA 2017: 17th International Conference, Trieste, Italy, July 3-6, 2017, Proceedings, Part VI 17*. Springer, 2017, pp. 203–217. [Online]. Available: https://doi.org/10.1007/978-3-319-62407-5_14
- [18] S. Li, H. Zhang, Z. Jia, C. Zhong, C. Zhang, Z. Shan, J. Shen, and M. A. Babar, "Understanding and addressing quality attributes of microservices architecture: A systematic literature review," *Information and software technology*, vol. 131, p. 106449, 2021. [Online]. Available: <https://doi.org/10.1016/j.infsof.2020.106449>
- [19] N. Alshuqayran, N. Ali, and R. Evans, "A systematic mapping study in microservice architecture," in *2016 IEEE 9th International Conference on Service-Oriented Computing and Applications (SOCA)*, 2016, pp. 44–51. [Online]. Available: <https://doi.org/10.1109/SOCA.2016.15>
- [20] V. Bushong, A. S. Abdelfattah, A. A. Maruf, D. Das, A. Lehman, E. Jaroszewski, M. Coffey,

- T. Cerny, K. Frajtak, P. Tisnovsky *et al.*, “On microservice analysis and architecture evolution: A systematic mapping study,” *Applied Sciences*, vol. 11, no. 17, p. 7856, 2021. [Online]. Available: <https://doi.org/10.3390/app11177856>
- [21] S. Lungu and M. Nyirenda, “Current trends in the management of distributed transactions in micro-services architectures: A systematic literature review,” *Open Journal of Applied Sciences*, vol. 14, no. 9, pp. 2519–2543, 2024. [Online]. Available: <https://doi.org/10.4236/ojapps.2024.149167>
- [22] R. Sarkis-Onofre, F. Catalá-López, E. Aromataris, and C. Lockwood, “How to properly use the prisma statement,” *Systematic Reviews*, vol. 10, pp. 1–3, 2021. [Online]. Available: <https://doi.org/10.1186/s13643-021-01671-z>
- [23] A.-W. Harzing, “Publish or Perish — harzing.com,” <https://harzing.com/resources/publish-or-perish>, 2016, [Accessed 06-04-2024].
- [24] T. Heyman, D. Preuveneers, and W. Joosen, “Scalar: Systematic scalability analysis with the universal scalability law,” in *2014 International Conference on Future Internet of Things and Cloud*. IEEE, 2014, pp. 497–504. [Online]. Available: <https://doi.org/10.1109/FiCloud.2014.88>
- [25] D. Monteiro, R. Gadelha, P. H. M. Maia, L. S. Rocha, and N. C. Mendonça, “Beethoven: an event-driven lightweight platform for microservice orchestration,” in *Software Architecture: 12th European Conference on Software Architecture, ECSA 2018, Madrid, Spain, September 24–28, 2018, Proceedings 12*. Springer, 2018, pp. 191–199. [Online]. Available: https://doi.org/10.1007/978-3-030-00761-4_13
- [26] L. Baresi and G. Quattrocchi, “Cocos: A scalable architecture for containerized heterogeneous systems,” in *2020 IEEE International Conference on Software Architecture (ICSA)*. IEEE, 2020, pp. 103–113. [Online]. Available: <https://doi.org/10.1109/ICSA47634.2020.00018>
- [27] L. L. Jiménez and O. Schelén, “Doema: A decentralized orchestrator for containerized microservice applications,” in *2019 IEEE Cloud Summit*. IEEE, 2019, pp. 45–51. [Online]. Available: <https://doi.org/10.1109/CloudSummit47114.2019.00014>
- [28] G. Márquez, M. M. Villegas, and H. Astudillo, “A pattern language for scalable microservices-based systems,” in *Proceedings of the 12th European Conference on Software Architecture: Companion Proceedings*, 2018, pp. 1–7. [Online]. Available: <https://doi.org/10.1145/3241403.3241429>
- [29] D. Gesvindr, J. Davidek, and B. Buhnova, “Design of scalable and resilient applications using microservice architecture in paas cloud,” in *ICSOFIT*, 2019, pp. 619–630. [Online]. Available: <https://doi.org/10.5220/0007842906190630>
- [30] Y. Zhang, Y. Gan, and C. Delimitrou, “ μ qsim: Enabling accurate and scalable simulation for interactive microservices,” in *2019 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*. IEEE, 2019, pp. 212–222. [Online]. Available: <https://doi.org/10.1109/ISPASS.2019.00034>
- [31] S. Henning and W. Hasselbring, “A configurable method for benchmarking scalability of cloud-native applications,” *Empirical Software Engineering*, vol. 27, no. 6, p. 143, 2022. [Online]. Available: <https://doi.org/10.1007/s10664-022-10162-1>
- [32] W. Hasselbring and G. Steinacker, “Microservice architectures for scalability, agility and reliability in e-commerce,” in *2017 IEEE International Conference on Software Architecture Workshops (ICSAW)*. IEEE, 2017, pp. 243–246. [Online]. Available: <https://doi.org/10.1109/ICSAW.2017.11>
- [33] N. Dragoni, I. Lanese, S. T. Larsen, M. Mazzara, R. Mustafin, and L. Safina, “Microservices: How to make your application scale,” in *Perspectives of System Informatics: 11th International Andrei P. Ershov Informatics Conference, PSI 2017, Moscow, Russia, June 27-29, 2017, Revised Selected Papers 11*. Springer, 2018, pp. 95–104. [Online]. Available: https://doi.org/10.1007/978-3-319-74313-4_8
- [34] Z. Wang, Y. Xia, C. Sun, and L. Cheng, “Research on microservice application performance monitoring framework and elastic scaling mode,” in *Journal of Physics: Conference Series*, vol. 1617, no. 1. IOP Publishing, 2020, p. 012048. [Online]. Available: <https://doi.org/10.1088/1742-6596/1617/1/012048>
- [35] K.-H. Chow, U. Deshpande, V. Deenadhayalan, S. Seshadri, and L. Liu, “Scad: Scalability advisor for interactive microservices on hybrid clouds,” in *Companion of the 2023 International Conference on Management of Data*, 2023, pp. 127–130. [Online]. Available: <https://doi.org/10.1145/3555041.3589718>
- [36] N. Agnihotri and A. K. Sharma, “Evaluating paas scalability and improving performance using scalability improvement systems,” *IJRET: International Journal of Research in Engineering and Technology eISSN*, pp. 2319–1163, 2014.
- [37] M. Rusek, G. Dwornicki, and A. Orłowski, “A decentralized system for load balancing of containerized microservices in the cloud,” in *Advances in Systems Science: Proceedings of the International Conference on Systems Science 2016 (ICSS 2016) 19*. Springer, 2017, pp. 142–152. [Online]. Available: https://doi.org/10.1007/978-3-319-48944-5_14
- [38] Y. Niu, F. Liu, and Z. Li, “Load balancing across microservices,” in *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*. IEEE, 2018, pp. 198–206. [Online]. Available: <https://doi.org/10.1109/INFOCOM.2018.8486300>
- [39] Y. Liang and Y. Lan, “Telbm: A task chain-based load balancing algorithm for microservices,” *Tsinghua Science and Technology*, vol. 26, no. 3, pp. 251–258, 2020. [Online]. Available: <https://doi.org/10.26599/TST.2019.9010032>
- [40] D. Müssig, R. Stricker, J. Lässig, and J. Heider, “Highly scalable microservice-based enterprise architecture for smart ecosystems in hybrid cloud environments,” in *International Conference on Enterprise Information Systems*, vol. 2.

- SCITEPRESS, 2017, pp. 454–459. [Online]. Available: <https://doi.org/10.5220/0006373304540459>
- [41] C.-C. Crecana and F. Pop, “Monitoring-based auto-scalability across hybrid clouds,” in *Proceedings of the 33rd Annual ACM Symposium on Applied Computing*, 2018, pp. 1087–1094. [Online]. Available: <https://doi.org/10.1145/3167132.3167248>
- [42] G. Yu, P. Chen, and Z. Zheng, “Microscaler: Automatic scaling for microservices with an online learning approach,” in *2019 IEEE International Conference on Web Services (ICWS)*. IEEE, 2019, pp. 68–75. [Online]. Available: <https://doi.org/10.1109/ICWS.2019.00023>
- [43] —, “Microscaler: Cost-effective scaling for microservice applications in the cloud with an online learning approach,” *IEEE Transactions on Cloud Computing*, vol. 10, no. 2, pp. 1100–1116, 2020. [Online]. Available: <https://doi.org/10.1109/TCC.2020.2985352>
- [44] M. Gotin, F. Lösch, R. Heinrich, and R. Reussner, “Investigating performance metrics for scaling microservices in cloudiot-environments,” in *Proceedings of the 2018 ACM/SPEC International Conference on Performance Engineering*, 2018, pp. 157–167. [Online]. Available: <https://doi.org/10.1145/3184407.3184430>
- [45] S. Merkouche and C. Bouanaka, “Tera-scaler for a proactive auto-scaling of e-business microservices,” pp. 448–455, 2023. [Online]. Available: <https://doi.org/10.5220/0012093500003538>
- [46] H. Ahmad, C. Treude, M. Wagner, and C. Szabo, “Smart hpa: A resource-efficient horizontal pod auto-scaler for microservice architectures,” in *2024 IEEE 21st International Conference on Software Architecture (ICSA)*. IEEE, 2024, pp. 46–57. [Online]. Available: <https://doi.org/10.1109/ICSA59870.2024.00013>
- [47] L. Bacchiani, M. Bravetti, S. Giallorenzo, M. Gabbrielli, G. Zavattaro, and S. P. Zingaro, “Proactive–reactive microservice architecture global scaling,” *Journal of Systems and Software*, vol. 220, p. 112262, 2025. [Online]. Available: <https://doi.org/10.1016/j.jss.2024.112262>
- [48] A. A. Khaleq and I. Ra, “Agnostic approach for microservices autoscaling in cloud applications,” in *2019 International Conference on Computational Science and Computational Intelligence (CSCI)*. IEEE, 2019, pp. 1411–1415. [Online]. Available: <https://doi.org/10.1109/CSCI49370.2019.00264>
- [49] Y. Gan, Y. Zhang, K. Hu, D. Cheng, Y. He, M. Pancholi, and C. Delimitrou, “Leveraging deep learning to improve the performance predictability of cloud microservices,” *arXiv preprint arXiv:1905.00968*, 2019. [Online]. Available: <https://doi.org/10.48550/arXiv.1905.00968>
- [50] F. Klinaku, M. Frank, and S. Becker, “Caus: An elasticity controller for a containerized microservice,” in *Companion of the 2018 ACM/SPEC International Conference on Performance Engineering*, 2018, pp. 93–98. [Online]. Available: <https://doi.org/10.1145/3185768.3186296>
- [51] M. Abdullah, W. Iqbal, A. Mahmood, F. Bukhari, and A. Erradi, “Predictive autoscaling of microservices hosted in fog microdata center,” *IEEE Systems Journal*, vol. 15, no. 1, pp. 1275–1286, 2020. [Online]. Available: <https://doi.org/10.1109/JSYST.2020.2997518>
- [52] L. M. Al Qassem, T. Stouraitis, E. Damiani, and I. A. M. Elfadel, “Proactive random-forest autoscaler for microservice resource allocation,” *IEEE Access*, vol. 11, pp. 2570–2585, 2023. [Online]. Available: <https://doi.org/10.1109/ACCESS.2023.3234021>
- [53] I. Prachitmutita, W. Aittinonmongkol, N. Pojjana-suksakul, M. Supattatham, and P. Padungweang, “Auto-scaling microservices on iaas under sla with cost-effective framework,” in *2018 Tenth International Conference on Advanced Computational Intelligence (ICACI)*. IEEE, 2018, pp. 583–588. [Online]. Available: <https://doi.org/10.1109/ICACI.2018.8377525>
- [54] M. Abdullah, W. Iqbal, J. L. Berral, J. Polo, and D. Carrera, “Burst-aware predictive autoscaling for containerized microservices,” *IEEE Transactions on Services Computing*, vol. 15, no. 3, pp. 1448–1460, 2020. [Online]. Available: <https://doi.org/10.1109/TSC.2020.2995937>
- [55] H. Ahmad, C. Treude, M. Wagner, and C. Szabo, “Towards resource-efficient reactive and proactive auto-scaling for microservice architectures,” *Journal of Systems and Software*, p. 112390, 2025. [Online]. Available: <https://doi.org/10.1016/j.jss.2025.112390>
- [56] A. A. Khaleq and I. Ra, “Intelligent autoscaling of microservices in the cloud for real-time applications,” *IEEE Access*, vol. 9, pp. 35 464–35 476, 2021. [Online]. Available: <https://doi.org/10.1109/ACCESS.2021.3061890>
- [57] M. B. Taha, Y. Sanjalawe, A. Al-Daraiseh, S. Fraihat *et al.*, “Proactive auto-scaling for service function chains in cloud computing based on deep learning,” *IEEE Access*, 2024. [Online]. Available: <https://doi.org/10.1109/ACCESS.2024.3375772>
- [58] N. H. Do, T. Van Do, X. T. Tran, L. Farkas, and C. Rotter, “A scalable routing mechanism for stateful microservices,” in *2017 20th Conference on Innovations in Clouds, Internet and Networks (ICIN)*. IEEE, 2017, pp. 72–78. [Online]. Available: <https://doi.org/10.1109/ICIN.2017.7899252>
- [59] H. Khazaei, R. Ravichandiran, B. Park, H. Bannazadeh, A. Tizghadam, and A. Leon-Garcia, “Elascale: Autoscaling and monitoring as a service,” *arXiv preprint arXiv:1711.03204*, 2017. [Online]. Available: <https://doi.org/10.48550/arXiv.1711.03204>
- [60] E. Casalicchio and V. Perciballi, “Auto-scaling of containers: The impact of relative and absolute metrics,” in *2017 IEEE 2nd International Workshops on Foundations and Applications of Self* Systems (FAS* W)*. IEEE, 2017, pp. 207–214. [Online]. Available: <https://doi.org/10.1109/FAS-W.2017.149>
- [61] P. Agarwal and J. Lakshmi, “Cost aware resource sizing and scaling of microservices,” in *Proceedings of the 2019 4th International Conference on Cloud Computing and Internet of Things*, 2019, pp. 66–74.

- [Online]. Available: <https://doi.org/10.1145/3361821.3361823>
- [62] T. Lin and A. Leon-Garcia, "Towards a client-centric qos auto-scaling system," in *NOMS 2020-2020 IEEE/IFIP Network Operations and Management Symposium*. IEEE, 2020, pp. 1–9. [Online]. Available: <https://doi.org/10.1109/NOMS47738.2020.9110450>
- [63] A. Bhole, "Enhancing performance and scalability in microservices," *Journal of Engineering and Applied Sciences Technology*, pp. 1–9, 09 2024. [Online]. Available: [https://doi.org/10.47363/JEAST/2024\(6\)E162](https://doi.org/10.47363/JEAST/2024(6)E162)
- [64] K. Hu, L. Wen, M. Xu, and K. Ye, "Msars: A meta-learning and reinforcement learning framework for slo resource allocation and adaptive scaling for microservices," *arXiv preprint arXiv:2409.14953*, 2024. [Online]. Available: <https://doi.org/10.48550/arXiv.2409.14953>
- [65] A. Cholomskis, O. Pozdniakova, and D. Mažeika, "Cloud software performance metrics collection and aggregation for auto-scaling module," in *Information and Software Technologies: 24th International Conference, ICIST 2018, Vilnius, Lithuania, October 4–6, 2018, Proceedings 24*. Springer, 2018, pp. 130–138. [Online]. Available: https://doi.org/10.1007/978-3-319-99972-2_10
- [66] A. Jindal, V. Podolskiy, and M. Gerndt, "Performance modeling for cloud microservice applications," in *Proceedings of the 2019 ACM/SPEC International Conference on Performance Engineering*, 2019, pp. 25–32. [Online]. Available: <https://doi.org/10.1145/3297663.3310309>
- [67] E. Casalicchio, "A study on performance measures for auto-scaling cpu-intensive containerized applications," *Cluster Computing*, vol. 22, no. 3, pp. 995–1006, 2019. [Online]. Available: <https://doi.org/10.1007/s10586-018-02890-1>
- [68] A. Kwan, J. Wong, H.-A. Jacobsen, and V. Muthusamy, "Hyscale: Hybrid and network scaling of dockerized microservices in cloud data centres," in *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2019, pp. 80–90. [Online]. Available: <https://doi.org/10.1109/ICDCS.2019.00017>
- [69] V. Podolskiy, A. Jindal, and M. Gerndt, "Multilayered autoscaling performance evaluation: can virtual machines and containers co-scale?" *International journal of applied mathematics and computer science*, vol. 29, no. 2, pp. 227–244, 2019. [Online]. Available: <https://doi.org/10.2478/amcs-2019-0017>
- [70] Y. M. Ramirez, V. Podolskiy, and M. Gerndt, "Capacity-driven scaling schedules derivation for coordinated elasticity of containers and virtual machines," in *2019 IEEE International conference on autonomic computing (ICAC)*. IEEE, 2019, pp. 177–186. [Online]. Available: <https://doi.org/10.1109/ICAC.2019.00029>
- [71] N. C. Coulson, S. Sotiriadis, and N. Bessis, "Adaptive microservice scaling for elastic applications," *IEEE Internet of Things Journal*, vol. 7, no. 5, pp. 4195–4202, 2020. [Online]. Available: <https://doi.org/10.1109/IIOT.2020.2964405>
- [72] C. K. Rudrabhatla, "A quantitative approach for estimating the scaling thresholds and step policies in a distributed microservice architecture," *IEEE Access*, vol. 8, pp. 180 246–180 254, 2020. [Online]. Available: <https://doi.org/10.1109/ACCESS.2020.3028310>
- [73] B. Choi, J. Park, C. Lee, and D. Han, "phpa: A proactive autoscaling framework for microservice chain," in *Proceedings of the 5th Asia-Pacific Workshop on Networking*, 2021, pp. 65–71. [Online]. Available: <https://doi.org/10.1145/3469393.3469401>
- [74] H. X. Nguyen, S. Zhu, and M. Liu, "Graph-phpa: graph-based proactive horizontal pod autoscaling for microservices using lstm-gnn," in *2022 IEEE 11th International Conference on Cloud Networking (CloudNet)*. IEEE, 2022, pp. 237–241. [Online]. Available: <https://doi.org/10.1109/CloudNet55617.2022.9978781>
- [75] Y. Kim, J. Park, J. Yoon, and J. Kim, "Improved q network auto-scaling in microservice architecture," *Applied Sciences*, vol. 12, no. 3, p. 1206, 2022. [Online]. Available: <https://doi.org/10.3390/app12031206>
- [76] M. ZargarAzad and M. Ashtiani, "An auto-scaling approach for microservices in cloud computing environments," *Journal of Grid Computing*, vol. 21, no. 4, p. 73, 2023. [Online]. Available: <https://doi.org/10.1007/s10723-023-09713-7>
- [77] V. M. Mostofi, E. Krul, D. Krishnamurthy, and M. Arlitt, "Trace-driven scaling of microservice applications," *IEEE Access*, vol. 11, pp. 29 360–29 379, 2023. [Online]. Available: <https://doi.org/10.1109/ACCESS.2023.3260069>
- [78] J. Santos, T. Wauters, B. Volckaert, and F. De Turck, "gym-hpa: Efficient auto-scaling via reinforcement learning for complex microservice-based applications in kubernetes," in *NOMS 2023-2023 IEEE/IFIP Network Operations and Management Symposium*. IEEE, 2023, pp. 1–9. [Online]. Available: <https://doi.org/10.1109/NOMS56928.2023.10154298>

Citation: N. Gurung S. Shrestha and R. Chulyadyo. *Scalability in Microservices: A Systematic Literature Review*. Journal of Computer Science & Technology, vol. 25, no. 2, pp. 128–143, 2025.

DOI: 10.24215/16666038.25.e11.

Received: December 26, 2025 **Accepted:** May 8, 2025.

Copyright: This article is distributed under the terms of the Creative Commons License CC-BY-NC-SA.