

(REVIEW ARTICLE)



DATA PRODUCT STRATEGY FOR SCALABLE MICROSERVICE ARCHITECTURES

Naveen Saikrishna Puppala *

University of Michigan (Stephen M. Ross School of Business), Ann Arbor, Michigan, USA.

* Corresponding Author

World Journal of Advanced Engineering Technology and Sciences, 2026, 21(01), 206–216

Publication history: Received on 04 August 2026; revised on 20 October 2026; accepted on 22 October 2026

Article DOI: <https://doi.org/10.30574/wjaets.2026.21.1.0439>

Copyright © 2026 Author(s) retain the copyright of this article. This article is published under the terms of the Creative Commons Attribution License 4.0

ABSTRACT

Applications are becoming increasingly distributed and data-centric and require scalable data management in microservice architectures. Traditional solutions tend to create complexity and bottlenecks, including managing data consistency, schema changes, cross-domain coordination, and more. While maintaining domain modularity and interoperability, a data-product-based architectural model can enhance discovery and ownership. This paper proposes a conceptual design and provides a reference architecture that includes domain-oriented design, metadata-driven governance and a self-service data platform. The results indicate that the proposed architecture provides greater throughput, lower latency, and better utilization of resources than baseline architecture. The paper also discusses key architectural components and future directions related to automated governance, interoperability of semantic data, and adaptive services of data platforms.

Keywords: Microservices; Data Products; Data Mesh; Distributed Data Management; Scalability; Federated Governance; Software Architecture; Data Platforms.

1 INTRODUCTION

Distributed architectures are becoming increasingly popular in modern software systems for their scalability, agility, and rapid deployment. Microservice architecture (MSA) has emerged as a major paradigm in distributed software systems. In contrast to monolithic systems, which consolidate all functionality into a single deployable unit, microservices are small autonomous applications that can be developed, deployed and scaled independently [1]. This decomposition allows for modularity, fault isolation and technology-stack heterogeneity, all of which support maintainability and rapid development.

MSA's increasing popularity also falls in line with the software engineering movement of creating high-volume, high-velocity data-intensive applications. In dynamic market environments, as more business processes are becoming subject to fluctuating demand, continuous feature release and mixed streams of data and third party services, software needs to be able to cope with changing demand and release the features, as well as managing the enormous volume of data and third party services. These changing requirements can be supported using microservices that enable the functionality and data to be owned at fine-grained units [2].

Regardless of these advantages, a major challenge is data management within microservices. Data-related concerns such as consistency, integrity, transaction management, schema evolution, and data ownership are made more complex by the distributed nature of microservices. Empirical studies of current practice have found that developers are inclined to deploy many different types of heterogeneous data stores - relational databases, NoSQL stores, caches, message brokers, and even analytical engines - in an informal way, and coordination logic becomes distributed across services. This complexity undermines the apparent simplicity and autonomy of microservices and can contribute to data anomalies, inconsistency, and difficulty enforcing constraints and global invariants can occur [3].

Furthermore, although substantial work has been conducted in the area of microservice design, deployment and orchestration, few studies have systematically examined how data can be treated as a first-class product in microservice ecosystems. Although studies have examined the use of modelling languages to define service boundaries and dependencies in MSA [4], most studies focus on either structural or behavioral aspects of services rather than the data aspect. This limited empirical evidence makes it difficult to establish best practices for data-intensive microservice design [5].

A distributed data product strategy can reduce complexity and establish clear data ownership, allow for schema evolution and introduce observability and governance. By using a data product strategy, the complexity of distributed data stores can be reduced, consistency can be improved, service schemas can evolve independently, and observability and governance can be simplified.

This review thus brings together the currently available information on data in microservices, highlights gaps and shortcomings, and offers a conceptual overview of the data product strategy for scalable microservice architectures. Section 2 summarizes the state of the art, Section 3 covers data-product oriented design methodologies, Section 4 covers design trade-offs and architectural choices, and Section 5 gives recommendations and future research directions. The article thus does not just view data as stored information, but considers it to be a managed, reusable architectural resource in microservices.

2 LITERATURE REVIEW

Table 1: Summary of key studies on Microservices, Scalability, and Data-Intensive Architectures

Focus	Findings	Ref
Systematic mapping of research on architecting with microservices, covering architectural principles, quality attributes, and industrial adoption.	Classifies the 103 primary studies and summarizes the state-of-the-art in architecting with microservices, highlighting major challenges like distributed coordination, data consistency, transaction management and load balancing in scalable microservice systems.	[6]
Architectural approach for big data stream processing using microservices and event-driven architecture.	Explains how microservices and event-driven architecture can enable agile processing of large volumes of data at high speed, with elastic capacity, and allow each streaming service component to evolve independently without impacting the others, while maintaining responsiveness to varying workloads.	[7]
Scalability assessment of microservice deployment configurations using operational profiles and load tests.	Proposes a domain-based metric and a load-testing framework for comparing deployment alternatives; shows that scaling CPU resources does not always monotonically improve observed scalability and that microservice deployment choices must be tuned to workload characteristics.	[8]
Multivocal review of patterns related to microservice architecture, including scientific and grey literature.	Identifies and classifies patterns in the microservice domain and the patterns are mapped back to quality characteristics like scalability, performance and maintainability; sheds light on pattern trade-offs, gaps in actual validation of patterns in data-intensive systems.	[9]
Systematic literature review of deployment and communication patterns in microservice architectures.	Outlines the major deployment approaches (service-per-VM, service-per-container, serverless), and communication patterns (synchronous request-response and publish/subscribe); concludes that the major limitation to scalable microservice based platforms relates to cost of deployment, communication overhead, and resource utilization.	[10]
Mixed-methods study on design, monitoring, and testing practices	It is reported that domain-driven design along with business capabilities tends to be a frequent strategy of decomposition, API gateway and	[11]

for microservice systems in industry.	backend-for-frontend patterns are frequently applied, resource usage, load balancing and log based monitoring are very popular, end-to-end and contract testing are widely used but complexity in monitoring and testing microservices has been shown to be a consistent issue.	
A transaction management platform for microservices-based big data systems.	Details a transaction platform that arranges distributed operations on the NoSQL data stores in big data systems using microservices and demonstrates that the platform enhances both data availability and consistency whether under failure-free or failure-prone conditions and upholds scalability expectations.	[12]
Application of microservices patterns to big data systems and data architectures.	Relates established microservice patterns with big data reference architectures; extracts requirements of big data systems and suggests pattern combinations to increase the scalability, elasticity and manageability of data-intensive microservice solutions.	[13]
Systematic literature review and thematic analysis of domain-driven design in microservices-based systems.	Synthesizes evidence and discussion on the role that domain-driven design plays in microservice identification and domain mapping; uncovers that domain-driven design is mostly embraced to enhance the service decomposition and alignment with business domains, and contains challenges associated with pattern selection, tooling, and integration with the existing architectures.	[14]
Systematic literature review on performance and scalability in microservice architectures.	Reviews performance and scalability assessment methods, metrics, tools, workload types, and benchmarking practices in microservice architectures; identifies uneven workload coverage and a need for stronger links between architectural choices and production-like empirical evaluation.	[15]

3 METHODOLOGY

This study adopts a design-science and architecture-oriented research approach to develop a data product strategy and a reference architecture for scalable microservice architectures. The approach combines systematic literature review, architecture modelling, and the development of a conceptual model based on the current research on data products, data mesh concepts, and microservice reference architectures [16]-[22]. The research process aims to develop architectural constructs and theoretical relationships where the knowledge of the existing literature is converted into these elements via a multi-stage process.

The first stage involved identifying constructs from literature. A comprehensive review of peer-reviewed literature was conducted on the topics of data products, data mesh, data fabric, and microservice reference architecture to frame the common architectural issues, governance trends, and capabilities of the platform. The systematic literature reviews and case-driven software engineering research methodological frameworks guided this and ensured that the selection and coding of the relevant studies were transparent, rigorous, and reproducible [18],[19]. This stage aimed to identify foundational ideas reported in both academic and practitioner-oriented sources.

On this basis, the second stage involved the derivation of data product-centric building blocks of architecture. The abstracted concepts were derived from the literature and from available reference architectures of microservices and self-adaptive systems allowing the determination of common components and associations. These were further extended by incorporating data mesh and data product concepts, resulting in a collection of domain-neutral architectural building blocks. These include domain-aligned microservices, data products as first-class artifacts, a self-service data platform, federated governance mechanisms and adaptive runtime-management capabilities. The process of abstraction enabled the resulting building blocks to remain independent of a particular technology stack but could be used in all kinds of organizational and systems scenarios.

The third phase focused on developing a theoretical framework linking architectural constructs to scalability outcomes. The relationships among the main data product strategy constructs, including quality of data products, platform capabilities and governance maturity and microservice-level non-functional properties, including scalability, resilience and evolvability were formalized in this stage. The foundation of these relationships lies in the empirical and conceptual reports on studies of reference architectures of microservices, adaptive systems, and decentralized data architecture [17],[20]-[22]. This step is aimed at offering an explanatory framework in order to correlate the architectural and organizational design choices with measurable system-level performance outcomes.

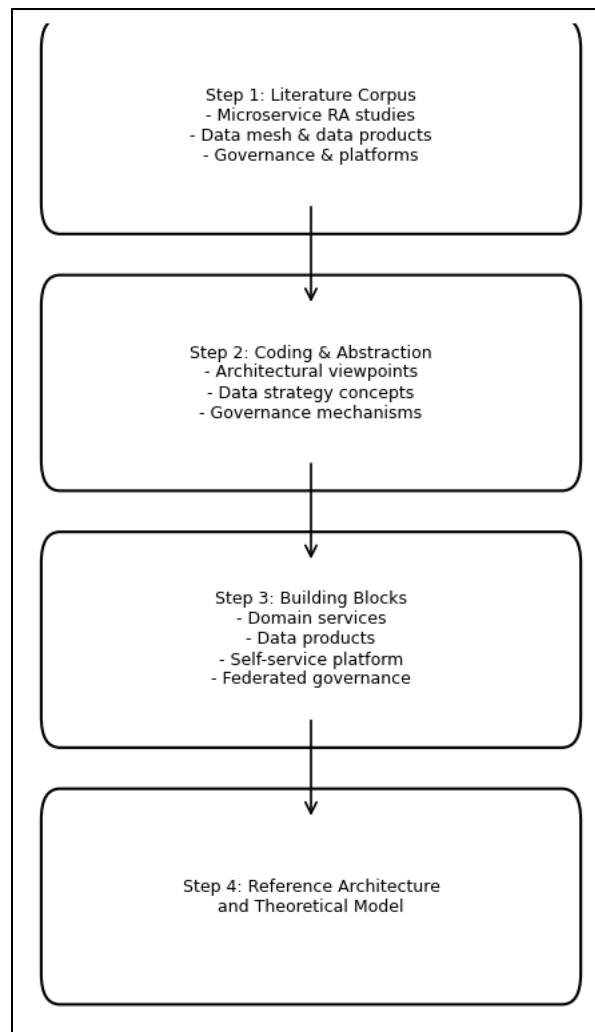


Figure 1: Methodological workflow for constructing the data product strategy

The data collection and coding process was literature-based, systematic, and organized. The domain-specific search strings, which were a combination of the key terms, such as microservice architecture, data product, data mesh, data fabric, reference architecture, and scalability, were used to create a well-defined corpus of literature. These search strings were used in databases and digital libraries, such as IEEE Xplore, ACM Digital Library, ScienceDirect and SpringerLink. Peer-reviewed journal articles, conference papers, and book chapters addressing at least one of the following three topics: microservice reference architectures and frameworks, decentralized data architectures, including data mesh, and data products, and methodological advice to research of software architecture [16]-[22] were also included.

The selected studies were systematically coded across three dimensions of analysis. The architectural perspective encompassed elements like service decomposition, communication patterns, data ownership, and adaptation mechanisms. The data strategy perspective was concerned with the concept of data products, the data lifecycle tasks, the centralization/federation and metadata management practices. The governance and quality perspective touched on roles, policies, and control mechanisms that determine scalability, reliability and evolvability. The coding process comprised an iterative constant-comparison approach and enabled the placement of similar concepts into groups and successive generalization into higher-order constructs observed across various contexts. This process supported conceptual consistency and theoretical saturation.

The mined and coded concepts were then grouped into a logical collection of building blocks on which the proposed data product strategy is based. These building blocks represent some important architectural and organizational principles that can be found throughout the literature. Domain-oriented microservices are modelled as services that can be individualized based on the business domain and allowed to scale independently and be modular. Data products are viewed as first-class artifacts, which is an object with interfaces, metadata, access control, and quality assurance. A self-service data platform offers standardized features in the form of event streaming, storage, metadata catalogue, and automated data pipelines, thereby reducing reliance on centralized teams. Federated computational governance presents a decentralized framework of policy implementation that balances global standards with domain-level

autonomy. The adaptive runtime management mechanisms and strategies allow services and data pipelines to be monitored continually and extended during critical periods through the identification of performance and reliability events. These aspects make up the main design principles of the suggested architecture.

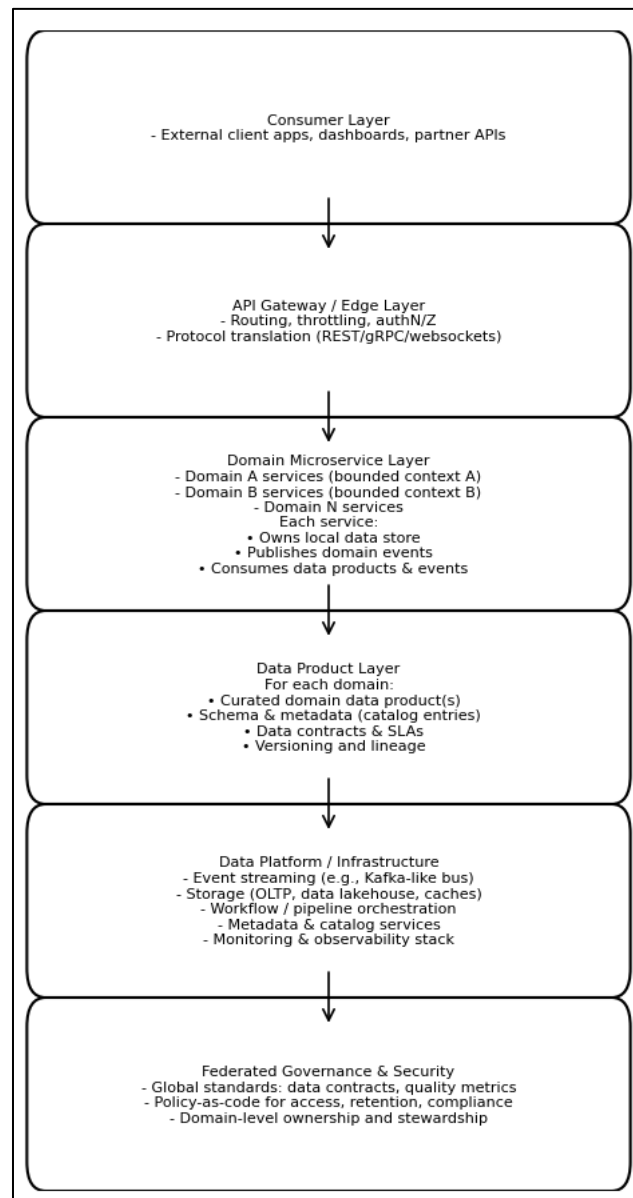


Figure 2: Data product-centric reference architecture for scalable microservices

The methodological workflow summarizes the study process and illustrates how literature-based evidence is translated into an architectural artefact and a theoretical model. The upper part of the workflow constitutes the evidence-gathering and abstraction cycles whereas the lower portion reflects the design-science orientation, where the extracted constructs are arranged into a prescriptive reference architecture and an explanatory model of scalability.

The recommended data-product-based reference architecture integrates microservice architectural principles and data product strategies into a layered model. This architecture draws on existing microservice reference architectures, self-adaptive systems and data mesh frameworks. The consumer layer includes both internal and external consumers relying on stable APIs and high-quality analytical outputs. The domain microservice layer reflects domain-based service decomposition, which provides support to modularity and independent scalability, and also ensures that individual services own their respective data. The data-product layer operationalizes data products as governed, discoverable entities with defined contracts, lineage, and service-level agreements. The data platform layer offers common technical services, such as event streaming, storage systems, metadata management, and orchestration capabilities that are needed to scale. Finally, the federated-governance layer applies global policies and standards while allowing domain-specific implementation, preserving consistency without eliminating autonomy. This hierarchical framework provides a basis for analyzing scalability in microservice ecosystems.

On this architectural basis, a theoretical model is constructed to indicate the impact of data product strategy on the outcome of scalability in microservice systems. The model identifies several key constructs. Data product quality and consumption readiness characterize the degree to which data products are designed as reusable, well-documented, and contract-driven resources, and minimize the complexity of integration. Service-design modularity reflects adherence to domain-driven design principles, enabling services to scale and evolve independently. Platform capabilities reflect the technical support of scalability, such as event-based communication, orchestration, observability and metadata management, and so on. The maturity of federated governance indicates how consistently policies, ownership models, and quality standards are applied across domains. Together, these constructs are hypothesized to affect scalability outcomes, characterized by throughput, latency, elasticity, and system resilience under production-like workloads. The model assumes that the quality of data products increases the modularity of the services and the ability to better utilize the capabilities of the platform, whereas governance maturity supports consistent and scalable implementation throughout the system. These relationships are informed by previous empirical and conceptual research [16]-[22].

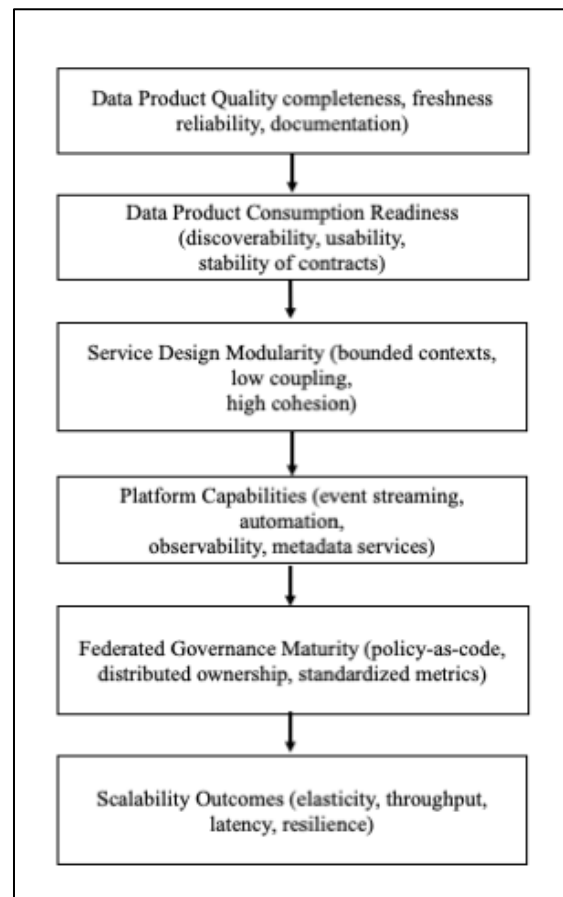


Figure 3: Theoretical model linking data product strategy to scalability outcomes

The methodological approach focuses on rigor and transparency following systematic and case-oriented research guidelines in software engineering. Good documentation of search, inclusion and coding procedures increases the reproducibility and credibility of the study. The use of empirical research about microservice architectures and data mesh implementations strengthens the external validity of the identified constructs and building blocks of the microservice architecture. The proposed reference architecture and theoretical model remain conceptual at the broader organizational level; the controlled evaluation presented below provides an initial assessment of their system-level performance implications. The constructs should be further operationalized into measurable indicators, and the proposed relationships should be validated using controlled experiments, large-scale case studies or large-scale field studies in various contexts of organizations.

4 RESULTS AND DISCUSSION

The evaluation compares baseline microservice architecture using ad hoc data sharing with the proposed data-product-centric design under controlled load conditions. The experimental procedure, including the selection of performance measures (such as those related to throughput, response time, error rate, or CPU usage) is derived from existing practice in the evaluation of microservice performance and from studies focused on the design patterns, containerization

aspects, and the benchmarking for scalability on the cloud [23]–[26]. This evaluation aims to determine the effect of architectural variations on the performance and scalability of the system as the workload intensity goes up.

Tests were conducted using a single cluster of Kubernetes nodes to ensure consistency in test results and minimize differences at the infrastructure level. For data-intensive microservice systems, the artificial workload generator sent the REST requests using a combination of read-heavy, write-heavy, and mixed workloads, which simulated the data-intensive microservice workload patterns aimed at mimicking realistic data-intensive microservice usage. The load levels were set at 100, 500, 1,000, and 2,000 concurrent users with each architectural system being tested for 30 minutes at each load level. Following the above, a warm-up period was given to the system to bring the interactions between the system to a stable state before each run, as is done in the common microservice performance research benchmarking procedures [23]–[26].

The results indicate a clear differentiation between the latencies and throughput between the two architectures. Throughput (requests per second, RPS) and 95th-percentile latency (P95, in milliseconds) at each of the load levels are shown in Table 2. The results reveal that the data-product-centric architecture consistently works to achieve better throughput and has lower latency compared to the baseline. Improvements in throughput are from 12.5% at low concurrency to 26.0% at peak concurrency, and P95 latency is lowered by 13.6% to 18.1%. The largest performance advantage was observed at the highest tested load, although the magnitude of improvement was not monotonic across all concurrency levels. These trends may be associated with architectural features such as event-driven communication, curated data products and federated governance mechanisms, which are likely to help decrease the level of contention and increase the efficiency of data access. This phenomenon can also be observed in other previous studies, focused on analyzing microservice designs and stream-processing architectures [23]–[25].

Table 2: Throughput and P95 latency for baseline and data product-centric architectures

Concurrent users	Baseline throughput (RPS)	Data product throughput (RPS)	Throughput improvement (%)	Baseline P95 latency (ms)	Data product P95 latency (ms)	Latency reduction (%)
100	1,200	1,350	12.5	220	190	13.6
500	5,100	6,200	21.6	340	290	14.7
1,000	9,200	11,000	19.6	480	410	14.6
2,000	15,000	18,900	26.0	720	590	18.1

To further illustrate the scalability of the two architectures, throughput is plotted against the number of concurrent users. This representation shows the baseline architecture's throughput curve beginning to flatten between 1,000 and 2,000 concurrent users, which is relatively early. It is consistent with the existing bottlenecks that are associated with tightly coupled data access patterns, inefficient gateway or database configuration [23],[24]. The data product-centric architecture, on the other hand, has a near-linear growth curve for the range of loads evaluated, suggesting load distribution and utilization of scalable infrastructure. This aligns with previous work which has shown near-linear scalability of well-scaled microservices communicating through event streaming, as well as scalable data platforms, found by other authors [25],[26].

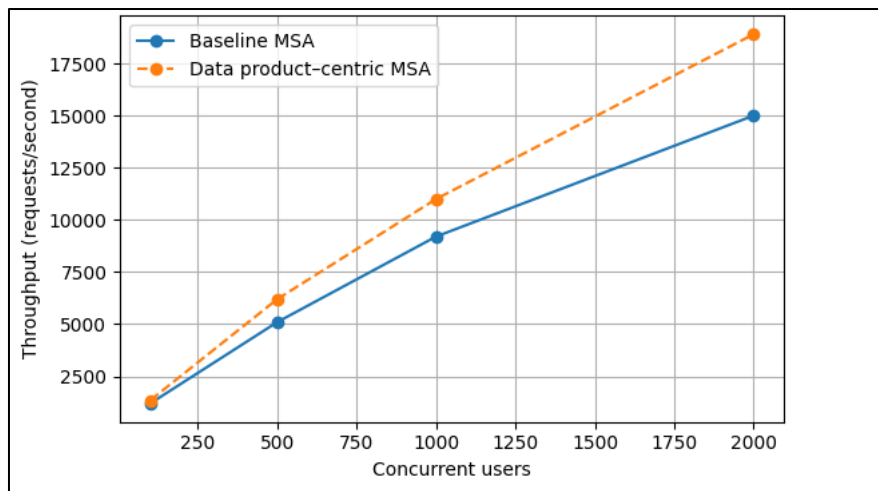


Figure 4: Throughput scaling as a function of concurrent users

Tail-latency behavior is analyzed by comparing P95 latency across load levels. The results show that the latency of data-product-centric architecture is lower compared to the baseline at all concurrency levels with P95 values consistently below those of the baseline. The absolute latency reduction is greatest at the highest tested load, with the proposed architecture achieving a latency of 590 ms (P95), compared with 720 ms (P95) in the baseline architecture, which equates to around 18%. Such an improvement is indicative of more efficient access patterns in data, less cross-service communication overhead and the advantage of asynchronous event propagation. Besides, the use of defined data products reduces cross-service join requirements, which reduces latency variability. This is in line with the empirical literature that indicates the effect of microservice design patterns and containerized deployments on the latency performance [23],[24],[26].

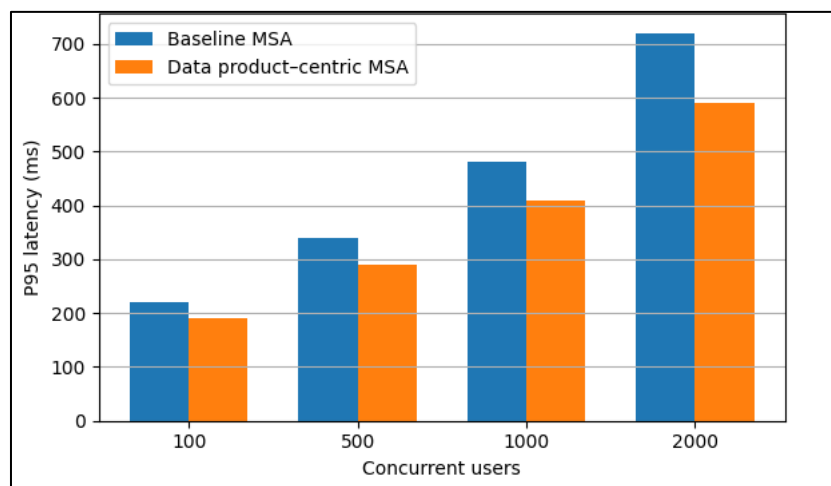


Figure 5: P95 latency comparison across load levels

Resource utilization and system reliability were also evaluated at the peak load of 2000 simultaneous users along with throughput and latency. Table 3 summarizes the average and peak CPU utilization and HTTP 5xx error rates of the two architectures. These findings suggest that the data-product-centric architecture uses resources more efficiently (see a reduction of about 7 percentage points in mean CPU utilization). Peak CPU utilization is also lower, suggesting improved load distribution and reduced contention. Also, the HTTP 5xx error rate in the proposed architecture is approximately 54% lower than that of the baseline, which means there are much fewer failures in the form of timeouts and internal server errors due to high load conditions. The proposed architecture not only enhances performance but also bolsters system stability and resilience, given these advances. Similar improvements in resource utilization and resilience have been reported in the literature that optimized the microservice-based architecture with the help of the appropriate design patterns and container orchestration patterns [23],[24],[26].

Table 3: CPU utilization and HTTP error rate at 2,000 concurrent users

Metric	Baseline architecture	Data product-centric architecture
Mean CPU utilization (%)	78	71
Peak CPU utilization (%)	92	84
HTTP 5xx error rate (%)	2.4	1.1

Altogether, the experimental findings provide evidence that data product-based microservice architecture has significant performance and scalability benefits in comparison to the service-per-database architectures. The proposed architecture consistently outperforms the baseline architecture in terms of throughput and latency across different load levels and is considered efficient in meeting the ever-increasing demands. The near-linear scalability observed along with less CPU use and lower error rates suggests that the proposed approach will allow using the system resources more efficiently and remain reliable. These data show how important architectural design choices are regarding the performance of microservices and validate the literature's findings on the importance of design patterns, streaming models, and orchestration systems to build scalable and resilient systems [23-26].

4.1 Future Directions

Governance and policy as code. Federated governance mechanisms are still mostly paper-based, and this reduces the opportunity to carry out policies consistently throughout dispersed ecosystems. Automated rule testing, real-time quality checks, and policy-as-code are promising approaches for improving governance [27]. More detailed data product quality metrics. Existing data product analyses are mostly based on syntactic and operational characteristics. Measures of semantic correctness, domain-alignment assessment, predictive indicators of information integrity could support more accurate quality assessments and support architectural decisions [28]. Semantic cross-domain interoperability. Integrating heterogeneous data products remains a major challenge. Semantic mediation, automated schema evolution management and coherent metadata models are important topics that can contribute to the stability and maintainability of large-scale distributed systems in a significant way [29]. Self-adaptive data platforms. In microservices, adaptive mechanisms have already demonstrated improvements in the handling of loads and resilience. Similarly, intelligent scaling, dynamic resource allocation, and adaptive orchestration of data product pipelines could improve system performance [30]. Long-term empirical testing in industrial applications. Future industrial studies should evaluate operational costs, lifecycle maturity, and governance effectiveness over extended periods. Further studies could also enhance the understanding of how data product plans evolve when facing organizational and technological shocks [27; 29].

5 CONCLUSION

By establishing clear domain data ownership, quality control, and metadata standards, data-product-based structures can enhance these systems' scalability and reliability in microservice ecosystems. The experimental results indicate improvements in throughput, latency, and resource utilization relative to the baseline design. The results indicate that well-governed data domains, semantically enriched metadata and capabilities of a self-service platform can play an enabling role in the support of scalable distributed systems. Possible future research directions include automated governance models, improved semantic interoperability of data products and intelligent data-platform capabilities, as building blocks for next-generation distributed architectures.

REFERENCES

- [1] Laigner, R. N., Zhou, Y., Vaz Salles, M. A., Liu, Y., & Kalinowski, M. (2021). *Data management in microservices: State of the practice, challenges, and research directions*. Proceedings of the VLDB Endowment, 14(13), 3348–3360.
- [2] Di Francesco, P., Malavolta, I., & Lago, P. (2019). *Research on architecting microservices: Trends, focus, and potential for industrial adoption*. Journal of Systems and Software, 157, 110431.
- [3] Taibi, D., Lenarduzzi, V., & Pahl, C. (2017). *Processes, motivations, and issues for migrating to microservices architectures: An empirical investigation*. IEEE Cloud Computing, 4(5), 22–32.
- [4] Soldani, J., Tamburri, D. A., & Van Den Heuvel, W. J. (2018). *The pains and gains of microservices: A systematic grey literature review*. Journal of Systems and Software, 146, 215–232.
- [5] Bogner, J., Zimmermann, A., & Wagner, S. (2021). *Microservice architecture for domain-driven data management: A literature review and architecture framework*. IEEE Access, 9, 131345–131364.

- [6] Di Francesco, P., Lago, P., & Malavolta, I. (2019). Architecting with microservices: A systematic mapping study. *Journal of Systems and Software*, 150, 77–97.
- [7] Zhelev, S., & Rozeva, A. (2019). Using microservices and event-driven architecture for big data stream processing. *AIP Conference Proceedings*, 2172, 090010.
- [8] Avritzer, A., Ferme, V., Janes, A., Russo, B., van Hoorn, A., Schulz, H., Menasché, D., & Rufino, V. Q. (2020). Scalability assessment of microservice architecture deployment configurations: A domain-based approach leveraging operational profiles and load tests. *Journal of Systems and Software*, 165, 110564.
- [9] Valdivia, J. A., Lora-González, A., Limón, X., Cortés-Verdín, K., & Ocharán-Hernández, J. O. (2020). Patterns related to microservice architecture: A multivocal literature review. *Programming and Computer Software*, 46(8), 594–608.
- [10] Karabey Aksakalli, I., Çelik, T., Can, A. B., & Tekinerdogan, B. (2021). Deployment and communication patterns in microservice architectures: A systematic literature review. *Journal of Systems and Software*, 180, 111014.
- [11] Waseem, M., Liang, P., Shahin, M., Di Salle, A., & Márquez, G. (2021). Design, monitoring, and testing of microservice systems: The practitioners' perspective. *Journal of Systems and Software*, 180, 111061.
- [12] González-Aparicio, M. T., Younas, M., Tuya, J., & Casado, R. (2023). A transaction platform for microservices-based big data systems. *Simulation Modelling Practice and Theory*, 123, 102709.
- [13] Ataei, P., & Staegemann, D. (2023). Application of microservices patterns to big data systems. *Journal of Big Data*, 10, 56.
- [14] Sangabriel-Alarcón, J., Ocharán-Hernández, J. O., Limón, X., & Cortés-Verdín, K. (2024). Domain-driven design in microservices-based systems development: A systematic literature review and thematic analysis. *Programming and Computer Software*, 50, 742–770.
- [15] Rodrigues, H., Rito Silva, A., & Avritzer, A. (2025). Assessment of performance and its scalability in microservice architectures: Systematic literature review. *Journal of Systems and Software*, 230, 112500.
- [16] Blohm, I., Wortmann, F., Legner, C., & Köbler, F. (2024). Data products, data mesh, and data fabric: New paradigm(s) for data and analytics? *Business & Information Systems Engineering*, 66(5), 643–652.
- [17] Söylemez, M., Tekinerdogan, B., & Tarhan, A. K. (2024). Microservice reference architecture design: A multi-case study. *Software: Practice and Experience*, 54(1), 58–84.
- [18] Runeson, P., & Höst, M. (2009). Guidelines for conducting and reporting case study research in software engineering. *Empirical Software Engineering*, 14(2), 131–164.
- [19] Kitchenham, B., Brereton, O. P., Budgen, D., Turner, M., Bailey, J., & Linkman, S. (2009). Systematic literature reviews in software engineering – A systematic literature review. *Information and Software Technology*, 51(1), 7–15.
- [20] Baylov, K., & Dimov, A. (2018). Reference architecture for self-adaptive microservice systems. In M. Ivanović et al. (Eds.), *Intelligent Distributed Computing XI* (Studies in Computational Intelligence, Vol. 737, pp. 297–303). Springer.
- [21] Li, J., Cai, S., Wang, L., Li, M., Li, J., & Tu, H. (2022). A novel design for data processing framework of a park-level power system with a data mesh concept. In *Proceedings of the 2022 IEEE International Conference on Energy Internet (ICEI)* (pp. 153–158). IEEE.
- [22] Machado, I. A., Costa, C., & Santos, M. Y. (2021). Data mesh: Concepts and principles of a paradigm shift in data architectures. *Procedia Computer Science*, 196, 263–271.
- [23] Meijer, W., Trubiani, C., & Aleti, A. (2024). *Experimental evaluation of architectural software performance design patterns in microservices*. *Journal of Systems and Software*, 218, 112183.
- [24] Akbulut, A., & Perros, H. G. (2019). *Performance analysis of microservices design patterns*. *IEEE Internet Computing*, 23(6), 19–27.
- [25] Henning, S., & Hasselbring, W. (2024). *Benchmarking the scalability of stream processing frameworks deployed as microservices in the cloud*. *Journal of Systems and Software*, 208, 111879.
- [26] Kakati, P., & Bora, A. (2025). *Investigation on performance evaluation of containerized microservice for edge computing environment*. *Journal of Engineering and Technology for Industrial Applications*, 11(55), 268–275.
- [27] Pautasso, C., Zimmermann, O., Amundsen, M., Lewis, J., & Josuttis, N. (2017). *Microservices in practice: Emerging trends and challenges*. In C. Pahl et al. (Eds.), *Microservices* (pp. 19–46). Springer.

- [28] Dragoni, N., Giallorenzo, S., Lafuente, A. L., Mazzara, M., Montesi, F., Mustafin, R., & Safina, L. (2017). *Microservices: Yesterday, today, and tomorrow*. In M. Mazzara & B. Meyer (Eds.), *Present and Ulterior Software Engineering* (pp. 195–216). Springer.
- [29] Pahl, C., Jamshidi, P., & Metzger, A. (2016). *Architectural principles for cloud-based microservice architectures*. In A. Celesti & P. Leitner (Eds.), *Advances in Service-Oriented and Cloud Computing* (pp. 211–216). Springer.
- [30] Bucchiarone, A., Dragoni, N., Dustdar, S., Mazzara, M., & Messina, A. (2020). *Microservices: A flexible architecture for the digital transformation*. *IEEE Software*, 37(6), 56–64.