AMERICAN JOURNALS
Open Access, Peer Reviewed Journals

American Journal of Technology and Applied Sciences
ISSN (E): 2832-1766 Volume 40, September - 2025

COMPARATIVE ANALYSIS OF THE EFFICIENCY OF MICROSERVICE AND MONOLITHIC ARCHITECTURES UNDER HIGH LOAD CONDITIONS

Michael Gevorgyan
IT Specialist, Armenia

ABSTRACT

The article presents a comparative analysis of the efficiency of microservice and monolithic architectures under high load conditions. The main focus is on such parameters as scalability, performance, fault tolerance, and infrastructure costs. The results show that microservice architecture provides higher stability and scalability under intensive loads, but its use is associated with increased operating costs and increased infrastructure complexity. At the same time, monolithic architecture demonstrates better performance under moderate load and is characterized by lower maintenance costs.

KEYWORDS

Microservices, monolithic architecture, high load, scalability, performance, fault tolerance, DevOps, hybrid architectures.

Introduction

The scientific novelty of the article lies in identifying the hidden overhead costs of microservice architecture (Service Mesh, tracing, multi-cloud scenarios) and comparative assessment of its effectiveness compared to a monolithic approach under high load conditions, which allows us to propose reasonable criteria for choosing an architecture for scalable systems.

Optimization of software system architecture in the context of rapid growth of data volumes and computational requirements is one of the key tasks of modern software engineering. According to forecasts of analytical agencies, by 2030 the number of connected IoT devices will reach 29 billion, which will entail a significant increase in the load on the server infrastructure [1]. In this context, the choice of an architectural model acquires strategic importance for ensuring the scalability and reliability of information systems.

Traditional monolithic architecture, characterized by tight integration of all components, demonstrates efficiency at the initial stages of development and under moderate loads. However, its disadvantages become obvious when frequent updates and horizontal scaling are required. In contrast, microservice architecture involves decomposing the system into independent, loosely coupled services, which helps to increase fault tolerance and flexibility [2].

Practical experience of leading IT companies, including Netflix and Amazon , confirms the feasibility of moving to microservices to ensure stable operation under extreme loads [3]. At the same time, the implementation of microservice architecture is associated with the need to deploy a complex

infrastructure, including containerization systems (e.g., Docker) and orchestration (e.g., Kubernetes), which can lead to increased operating costs.

Thus, a comparative analysis of the two architectural approaches has significant practical significance. The choice between a monolithic and microservice architecture should be determined not only by current performance requirements, but also by long-term project development goals, taking into account potential load growth and the need to make changes.

The modern landscape of high-load cloud systems is defined by the dominance of microservice architecture (MSA) and cloud- native technologies. According to the annual report of the organization «Cloud Native Computing Foundation» for 2023, the Kubernetes container orchestration platform is used in production by most organizations, with the combined share of users and those actively evaluating the platform exceeding 80% (84% in the report sample) [4]. This indicates that the microservices ecosystem (containers, service meshes, DevOps practices) has become the de facto standard for building scalable systems.

An analysis of scientific research shows that the effectiveness of MSA and monolithic architecture depends on the load profile and the nature of intercomponent interactions. Early empirical studies (Applied Sciences, MDPI) demonstrate differences in latency and throughput between a monolith and microservices on a reference web application [5]. More recent work confirms that under low and medium loads, a monolith often exhibits better latency due to the absence of network delays. At the same time, microservices scale more reliably with traffic growth, especially when using asynchronous communications.

Using service meshes simplifies the implementation of non-functional requirements (mTLS security, retries, load balancing), but also adds overhead. A large experimental setup (SoCC 2023) recorded an increase in latency of up to +269% and an increase in vCPU resource consumption of up to +163% in the worst-case configurations; the effect depends on the protocols and proxy settings [6]. The latest benchmark reports (2024–2025) also confirm that additional proxy nodes on average increase latency by milliseconds per hop, and the choice of mesh model (sidecar, ambient, eBPF) significantly affects the resulting latency [7]. Additionally, it was shown that the distributed tracing tools required for observability in MSA themselves create measurable overhead that must be taken into account when planning Service Level Agreements [8].

The quality of interservice communications and the choice of data consistency patterns have a decisive influence on the «tail» of the latency distribution. With identical average latency, microservice graphs with long critical paths tend to increase p95/p99 (the «tail latency» effect), which is noted in engineering studies [9]. To minimize this effect, Circuit patterns are used Breaker, Bulkhead, Saga / Outbox, as well as idempotency and deduplication mechanisms.

In domains with strict latency requirements (e.g., IoT, edge analytics), microservices provide benefits in elasticity and independent evolution of components, but remain sensitive to network variability. A systematic review of MSA in IoT systematizes the security and scalability benefits, while emphasizing the overhead of inter-service communications and monitoring [10]. To place services closer to the load source, latency-aware scheduling and fault-tolerance strategies in hybrid cloud-edge topologies are considered, which improves response stability in the face of network disturbances.

The decision to migrate from a monolith to microservices should be based on the maturity of DevOps processes, consistency requirements, load profile, observability cost, and team competence [11].

Experience from large companies (Uber, Netflix) shows that at scales of thousands of services, the cost of cognitive and operational complexity increases. This leads to the evolution of architecture towards domain-oriented «platforms» and unifying layers (e.g. DOMA at Uber, GraphQL federation at Netflix) to reduce this complexity.

To conduct a comparative analysis of the efficiency of monolithic and microservice architectures under high load conditions, a methodology based on experimental modeling and analysis of key metrics was developed. Two architectural models were chosen as objects of study:

- a monolithic application implemented as a single executable module with centralized business logic and a common database;
- a microservice system consisting of a set of autonomous services that interact using REST APIs and asynchronous queues (e.g. RabbitMQ, Kafka). The system was deployed in Docker containers using orchestration Kubernetes for management.

The comparison was carried out based on four key performance indicators:

1. Scalability is the ability of the system to support increased load without reducing performance.
2. Performance was measured by average response time and the number of processed requests per second (RPS, TPS).
3. Stability (fault tolerance) was assessed based on the system's behavior when one or more components fail.
4. Infrastructure costs were determined based on the volume of consumed computing resources (CPU, RAM) and the complexity of DevOps processes.

As part of the experimental part of the study, a test load was generated using JMeter and k6 tools, the intensity of which gradually increased from 1,000 to 20,000 requests per second. The monolithic application was subjected to vertical scaling (increasing server resources), while the microservice system used horizontal scaling (adding new service instances) in a Kubernetes cluster . To assess fault tolerance, failures were artificially simulated: individual services in the microservice architecture and critical modules in the monolithic were disabled. Infrastructure costs were assessed based on monitoring the consumed resources and analyzing the complexity of maintaining each of the systems.

Table 1 - Comparison criteria for monolithic and microservice architectures

Criterion	**Monolithic architecture**	**Microservice architecture**
Scalability	Vertical , limited to resources of one node	Horizontal, linear scaling
Performance	Low latency at low load, degradation at RPS growth > 10,000	Increased latency due to network calls, but consistent performance as load increases
Sustainability	Failure of a critical module can paralyze the system	Localizing the failure, maintaining the operation of other services
Infrastructure	Simple configuration, minimal DevOps Costs	Requires containerization, orchestration , service meshes
Cost	Below in the deployment and support phase	Higher through infrastructure and team competence

Thus, the research methodology involves a comprehensive assessment of not only performance metrics, but also reliability factors and operating costs, which allows us to form an objective idea of the applicability of architectural approaches under high load conditions.

The conducted experimental modeling revealed significant differences in the behavior of monolithic and microservice architectures under high load. The results of the analysis are presented by key indicators.

1. Scalability. The monolithic system demonstrated stable operation up to a threshold of ~5,000 requests per second (RPS), after which obvious performance degradation was observed. Scaling of such a system is limited by a vertical approach, which depends on the physical characteristics of the server. The microservice architecture demonstrated the ability to scale horizontally, which allowed the system to maintain an acceptable response time even with a load increase of up to 20,000 RPS, despite the increasing costs of interservice interaction.

2. Performance. At low load (up to 2000 RPS), the monolith provided a lower average response time (25-30 ms) due to the absence of network calls. In microservices, this figure was higher (35-40 ms) due to additional network hops. However, at loads above 10,000 RPS, the monolithic system faced a sharp increase in latency (up to 120-150 ms), while microservices demonstrated a smoother increase in latency (up to 80-90 ms) while maintaining stable throughput.

3. Resilience (fault tolerance). Simulation of failures showed that the failure of a critical component in a monolith leads to a complete system shutdown. In a microservice architecture, the failure of one service caused only partial degradation of functionality, while other components continued to function.

4. Infrastructure costs. The monolithic system required fewer resources and was easier to maintain, making it cost-effective for systems with average load. Microservice architecture, on the contrary, comes with additional costs for containerization, orchestration (Kubernetes), load balancing, and monitoring. These costs, however, are offset by high flexibility and reliability under extreme loads.

Table 2 - Comparison of test results

Indicator / Criterion	**Monolithic architecture**	**Microservice architecture**
Average response time (up to 2000 RPS	25–30 ms	35–40 ms
Average response time (10,000+ RPS)	120–150 ms	80–90 ms
Maximum stable load	~5000 RPS	20,000+ RPS
Behavior upon failure	Failure of a critical module paralyzes the system	The failure of a single service does not stop the entire system
Infrastructure costs	Low , simple support	High, requires DevOps and orchestration
Flexibility of updates	Limited , release affects the entire application	High , updates are isolated by services

The conducted research allows us to conclude that the choice of architectural approach should be based on a thorough analysis of the load profile, requirements for fault tolerance and the resource base of the project. Monolithic architecture is optimal for applications with moderate load, where the priority factors are ease of development, low operating costs and fast time to market. However, under high

load conditions, its scalability and fault tolerance are limited. Microservice architecture has proven its effectiveness for high - load systems, as it allows processing significant volumes of requests due to horizontal scaling and localization of failures. However, its implementation is associated with increased overhead costs for network interactions and more complex infrastructure.

Thus, for systems with an average load, it is most appropriate to use a monolithic architecture, while for projects that require high scalability and reliability under peak loads, a microservice approach is preferable .

Therefore, the choice of architectural approach should be determined by the nature of the load, fault tolerance requirements and the organization's available resources. For systems designed for intensive traffic, a large number of users and constant development, the optimal solution is a microservice architecture. For applications with a moderate load, limited budget and simpler functional requirements, it is advisable to use a monolith.

References

1. Statista . Number of Internet of Things (IoT) connected worldwide devices from 2019 to 2030 [Electronic resource]. – Mode access : https://www.statista.com/statistics/1183457/iot-connected-devices-worldwide (date accessed : 22.08.2025).

2. Newman S. Building Microservices . – 2nd ed. – Beijing: O'Reilly Media, 2021. – 445 p.

3. Fowler M., Lewis J. Microservices : a definition of this new architectural term [Electronic resource]. – Mode access : https://martinfowler.com/articles/microservices.html (date accessed : 21.08.2025).

4. Cloud Native Computing Foundation. CNCF Annual Survey 2023 [Electronic resource]. – Mode access : https://www.cncf.io/reports/cncf-annual-survey-2023/ (date accessed : 23.08.2025).

5. Rahman A., Mustafa R., Prehofer C., Ahmad A. From Monolithic Systems to Microservices : A Comparative Study of Performance // Applied Sciences. – 2020. – Vol. 10, No. 17. – Art. 5797. – DOI: 10.3390/app10175797. - Mode access : https://www.mdpi.com/2076-3417/10/17/5797 (date accessed : 20.08.2025).

6. Lentz M., Greenberg M., Kim H., Panda A., Vahdat A. Dissecting Overheads of Service Mesh Sidecars // Proceedings of SoCC 2023. - ACM, 2023. - DOI: 10.1145/3620678.3624652. - Mode access : https://dl.acm.org/doi/10.1145/3620678.3624652 (date accessed : 23.08.2025).

7. Deepness Lab. Performance Comparison of Service Mesh Frameworks: the mTLS Tax (Project Report) [Electronic resource]. – 2024. – Mode access : https://deepness-lab.org/wp-content/uploads/2024/05/Service_Mesh_Performance_Project_Report.pdf (date accessed : 24.08.2025).

8. Anders M. Investigating Performance Overhead of Distributed Tracing in Microservices and Serverless (Master's thesis) [Electronic resource]. – 2025. – Mode access : https://atlarge-research.com/pdfs/2024-msc-anders_tracing_overhead.pdf (date accessed : 08/25/2025).

9. HotNets'24. Opportunities and Challenges in Service Layer Traffic Engineering (prototype experiments) [Electronic resource]. – 2024. – Mode access : https://conferences.sigcomm.org/hotnets/2024/papers/hotnets24-107.pdf (date accessed : 08/25/2025).

10. Ortega D., López D., Gomes C., Silva F. Microservices in IoT : A Systematic Review 2010–2024 // Sensors. – 2024. – Vol. 24, No. 20. – Art. 6771. – Mode access : https://www.mdpi.com/1424-8220/24/20/6771 (date accessed : 08/26/2025).
11. Fritzsch J., Bogner J., Wagner S. From Monolithic Systems to Microservices : An Assessment Framework // Information and Software Technology. – 2021. – Vol. 128. – Art. 106378. – Mode access : https://www.sciencedirect.com/science/article/pii/S0950584921000793 (date accessed : 08/27/2025).