

THE SHIFT TO MICROSERVICES: ADVANTAGES AND CHALLENGES IN MODERN SOFTWARE ARCHITECTURE

Vinnytsia National Technical University

Анотація

У роботі досліджується перехід від монолітної архітектури до мікросервісної у сучасній розробці програмного забезпечення. Розглянуто основні причини зміни архітектурного підходу, проаналізовано переваги мікросервісів, зокрема масштабованість, гнучкість розробки, незалежність сервісів та ізоляцію відмов. Особливу увагу приділено проблемам, пов'язаним зі складністю управління розподіленими системами, забезпеченням узгодженості даних, тестуванням і безпекою. Також наведено порівняння з монолітною архітектурою та визначено умови доцільного використання мікросервісів.

Ключові слова: мікросервіси, програмна архітектура, масштабованість, розподілені системи, моноліт.

Abstract

This paper investigates the transition from monolithic architecture to microservices in modern software development. It examines the key reasons for this architectural shift and analyzes the main advantages of microservices, including scalability, development flexibility, service independence, and fault isolation. Special attention is given to challenges such as increased system complexity, data consistency, testing, and security issues. The paper also provides a comparison with monolithic architecture and identifies scenarios where microservices are most appropriate.

Keywords: microservices, software architecture, scalability, distributed systems, monolith.

Introduction

Over the past decade, software systems have significantly increased in complexity, scale, and functional requirements. Traditional monolithic architectures, where all components of an application are tightly integrated into a single codebase, were once sufficient for small and medium-sized systems. However, as applications grow, this approach leads to numerous limitations, including difficulties in scaling, long deployment cycles, and reduced flexibility in development.

To address these issues, microservices architecture has emerged as a modern alternative. In this approach, applications are divided into a collection of loosely coupled services, each responsible for a specific business function. These services communicate through lightweight protocols and can be developed, deployed, and scaled independently. This paradigm shift reflects the growing need for agility, resilience, and continuous delivery in modern IT environments [1, 2].

Research results

One of the most significant advantages of microservices architecture is scalability. In contrast to monolithic systems, where scaling requires replicating the entire application, microservices allow individual components to be scaled independently based on demand. This leads to more efficient resource utilization and better system performance under varying workloads.

Another important benefit is development flexibility. Different teams can work on separate services simultaneously, using different technologies, programming languages, and frameworks. This not only accelerates the development process but also enables organizations to adopt new technologies without rewriting the entire system.

Fault isolation is also a critical advantage. In a monolithic system, a single failure can potentially bring down the entire application. In contrast, microservices limit the impact of failures to specific services, improving overall system resilience and availability.

Despite these benefits, the transition to microservices introduces several challenges. One of the main issues is increased system complexity. Managing a distributed system requires advanced tools for service discovery, monitoring, logging, and orchestration [3]. Developers must handle inter-service communication, network latency, and potential failures in communication channels.

Data management is another complex aspect. In monolithic systems, data is typically stored in a single database, ensuring consistency. However, in microservices architecture, each service may have its own database, leading to challenges in maintaining data consistency across the system. Techniques such as eventual consistency and distributed transactions are often required, but they increase system complexity.

Testing and debugging also become more difficult in a microservices environment. Since the system consists of multiple interacting services, identifying the root cause of errors requires comprehensive monitoring and logging solutions. Integration testing becomes more complex compared to monolithic applications.

Security is another important concern. The presence of multiple services increases the number of entry points, making the system more vulnerable to attacks. Therefore, additional security measures such as authentication, authorization, and secure communication between services must be implemented.

In comparison with monolithic architecture, microservices provide greater flexibility and scalability but require more sophisticated infrastructure and management. Monolithic systems remain a viable option for smaller applications or projects with limited resources, where simplicity and ease of development are more important than scalability.

Conclusion

The shift to microservices architecture represents a significant evolution in modern software engineering. It enables organizations to build scalable, flexible, and resilient systems that can adapt to changing requirements. However, this approach also introduces new challenges related to complexity, data management, testing, and security.

Therefore, the decision to adopt microservices should be based on a careful analysis of project requirements, team expertise, and available resources. In many cases, a hybrid approach or gradual transition from monolithic architecture may be the most effective strategy.

REFERENCES

1. Microservices advantages and disadvantages URL: <https://www.ibm.com/think/insights/microservices-advantages-disadvantages>.
2. Advantages of microservices URL: <https://www.atlassian.com/microservices/cloud-computing/advantages-of-microservices>.
3. Design, Monitoring, and Testing of Microservices Systems: The Practitioners' Perspective URL: <https://arxiv.org/abs/2108.03384>

Ярован Денис Олександрович – студент групи ЗПІ-25б, Факультет інформаційних технологій та комп'ютерної інженерії, Вінницький національний технічний університет, м. Вінниця, e-mail: yarovan.dwnis@gmail.com.

Науковий керівник: **Чопляк Вікторія Володимирівна** – викладач англійської мови, кафедра іноземних мов, Вінницький національний технічний університет, м. Вінниця, e-mail: nikavnuchkova@gmail.com.

Yarovan Denys Oleksandrovych — student of group ЗПІ-25b, Faculty of Information Technology and Computer Engineering, Vinnytsia National Technical University, Vinnytsia, e-mail: yarovan.dwnis@gmail.com.

Scientific Supervisor: **Victoria V. Chopliak** – teacher of English, Foreign Languages Department, Vinnytsia National Technical University, Vinnytsia, e-mail: nikavnuchkova@gmail.com.