Microservices vs Monolith: A Comparative Analysis and Problem-Solving Approach in Web Development Area

Khant Hmue
University of Information Technology
Yangon, Myanmar
khanthmuu@uit.edu.mm

Myat Pwint Phyu
University of Information Technology
Yangon, Myanmar
myatpwintphyu@uit.edu.mm

Aye Myat Myat Paing
University of Information Tehcnology
Yangon, Myanmar
ayemyatmyatpaing@uit.edu.mm

2024 5th International Conference on Advanced Information Technologies (ICAIT) | 979-8-3315-1004-6/24/$31.00 ©2024 IEEE | DOI: 10.1109/ICAIT65209.2024.10754922

***Abstract*—With the rising complexity and advancement of web applications, many companies require an architecture that can adapt to their business requirements. This research highlights the pros, cons and appropriateness of monolithic, versus microservice architectures across various application types. Numerous studies suggest that microservice architecture offers benefits in scalability and maintenance compared to systems. The potential of microservices as a solution is explored in this study. An ecommerce web application case study employing microservice architecture is implemented to showcase the advantages of this approach. The aim of this paper is to demonstrate the advantages of utilizing microservices to address the limitations of monolithic systems by presenting a practical scenario using microservices. Overall, this paper provides an in-depth comparison between monolithic and microservices offering insights for stakeholders and developers making decisions, on web development project structures.**

Keywords—microservices architecture, monolithic architecture, web development

I. INTRODUCTION

In ever-growing web development area, architectural decisions made during the design and implementation of applications can make or break the scalability, maintainability, and overall performance of the systems with which they are tasked. Developing web applications using a monolithic architecture presents difficulties with maintainability, scalability, and agility. Their mutual dependency means that individual components can often be difficult to scale, rolling out upgrades without impacting the entire system. It is difficult and time-consuming in adding new additions and redeploys the entire system even in minor changes which leads to downtime. Finding bugs and maintaining codebase in monolithic architecture is quite challenging. Microservice addresses a number of challenges in these areas by breaking down large, integrated systems that come with monolithic architecture. Maintenance tasks become more straightforward, and utilizing independent services allows for faster deployment without having to take the full system down. It is also easier to scale out certain components and individual services work within a framework that makes teams more responsive and adaptable to changing business requirements. In the end, it offers greater scalability and flexible architectural solutions to traditional constraints, while traditional approaches to solving these problems allowed only limited scalability and flexibility across architectural solutions.

The objective of this paper is to perform a comprehensive and comparative analysis between microservice and monolithic architectures, while providing developers with actionable insights for effective problem solving. The structure of the paper is as follows. Section II offers a literature review on relevant research studies pertaining to both architectural approaches. Section III outlines the background theory by presenting fundamental concepts related to web development using these two architectures. Section IV offers the considerations of transition from monolithic to microservice architecture. In section V, implementation methodology details are explained regarding an ecommerce website that has been developed utilizing both architecture models. Next, in section VI, evaluation criteria such as performance testing has been carried out. The conclusion of the paper is presented in Section VII.

II. LITERATURE REVIEW

The transition from monolithic architectures to microservices is a widely debated topic in web development. Numerous studies have highlighted the advantages and challenges of these architectural styles aiming to offer insightful perspectives on their suitability for different usage situations and project requirements. Singh and Peddoju [1] in their research study developed a monolithic application and a microservice architecture to compare the two. During periods of high traffic, they tested both systems and discovered that the microservice design provide better performance in handling throughput compared to its counterpart based on their investigative results. One significant study by Fowler et al. (2014) [2] focused on microservice architecture highlighting its benefits in terms of scalability, maintainability and agility. The authors described how microservices enable teams to develop deployment and scale services independently leading to faster time to market and improvement in fault isolation.

In contrast Lewis and Fowler's research (2014) [3] examined into the challenges of transitioning from monolithic architectures to microservices. The authors discussed the complexities involved in breaking down monolithic applications into smaller services including issues related to communication and deployment automation. Additionally, O'Reilly (2018) survey [4] assessed the adoption trends and challenges faced by organizations implementing microservices. The utilization of microservice architecture has shown impressive results across several factors, including query response time, hosting expenses,

979-8-3315-1004-6/24/$31.00 ©2024 IEEE

hardware usage and packet-loss rate, thereby highlighting its advantages as a fast, independent and efficient service deployment option [5]. These outcomes reinforced the growing enthusiasm for adopting microservices due to their potential benefits in both scalability and agility.

In summary, there are a lot of researches and implementations of microservice and monolithic architectures for website development. Through the combination of academic studies and industry reports, this paper intends to provide developers and stakeholders with an extensive comparative evaluation as well as an effective problem-solving strategy designed to make informed decisions for choosing the right architecture.

III. Background

In this section, we explore the fundamental concepts of microservice and monolithic architectures in web development.

A. Monolithic Architecture

The monolithic architecture is a typical model for designing software, which aims through the binding of all components tightly together in a single structure. It has a single codebase therefore development time and the code are also made simpler. Enterprise applications commonly adhere to the classic three-tier model, comprising: commonly, web application consists of two things: (i) the "user interface" code that runs in the user's browser, such as HTML pages and JavaScript; (ii) server-side" business logic responsible for handling HTTP requests, executing domain logic, interacting with the database for data retrieval and updates, and generating HTML views for transmission to the browser; and (iii) the database backend. Being centralized, this approach shows its legitimacy by making very simple and convenient for small projects.

However, whilst software needs growing in terms of complexity, moving and scaling layers of monolithic systems can be challenging, often leading to slow deployment and lack of continuous delivery. Unlike the distributed applications that necessitate a complicated testing, deployment, debugging, and monitoring process, the simplicity of monolithic architecture - it is even simpler to understand - is its key benefit. Monolithic architecture allows the database, which is usually where external connections occur, to be located in one place and thus internal communications that make use of the intraprocess mechanisms are simplified. Nevertheless, as the application grows and its scope widens, the task of modification of the source code becomes more difficult. Such a problem can arise, when some of the changes made in one part of the application starts the errors in the other areas. The main disadvantage of this technology is the larger size of a monolithic architecture which results in a protracted startup time where development is slowed down and continuous deployment is impeded. With the application growing, the team is in difficulties to make the pattern of modules as it is impossible to set only the particular modules. Therefore, it is difficult to have the desired modular structure.

B. Microservice Architecture

Microservice architecture (MSA) is a software design style that breaks application into a group of small services. Each service can be deployed on its own. This type of design is used because it can be scaled easily, it is flexible and it can be managed. Mostly the cloud environments are typically associated with this style. In contrast with monolithic architectures – where all the components are tightly coupled together, microservices are smaller, freely deployed services which focus on particular business segments. Modularity is a focal point when it comes to advantages of microservices; this feature simplifies development, testing, and support purposes. Such steps can be taken simultaneously, where different teams involved adopt appropriate technology stack that can meet requirements of various functionalities. Each service can be deployed without any interference with other services. This modularity also works in line for continuous delivery and deployment processes which are a constituent of DevOps initiative where the attempt is to lessen development cycle size and provides the quick and certain software releases.

Conversely, microservices do not grow to adapt issues by itself, the developers have to adjust it. Private sector enterprises along with other institutions need to be provided with the answers to questions like security flaws, trust ability, and legal implications goals. Technical difficulties include guaranteeing the quality of performance, which can be effectively debugged, and keeping the consistency of the data in services. Additionally, thus a migration from a monolithic to a microservice architecture includes a specific course that might experience obstacles like organizational transformation and distributed monitoring [6].

In short, microservice architecture is a type of software design that remarkably improved agility and scalability. Even though there are issues related to that, but good planning and taking into account the specific needs of a project it can result in success and in exploiting the advantages of this modernized architectural style.

IV. Transition From Monolithic To Microservice

Even there are numerous benefits of microservice including scalability, modularity, and shorter deployment time, switching from monolithic to microservice architecture has some challenges to organizations. Organization demands new approaches for inter-service communication, service interaction and orchestration. The shift to a microservice introduces new problem of increased complexity that relates to the distributed services. Another challenge is the maintenance of the data consistency among distributed services. Transition also requires adapting to the type of tools and frameworks of containerization, orchestration, and monitoring. Transition to microservices require restructuring the development and operating teams into smaller, cross-functional teams. This change can face coordination services because distributed services still need to work together. This restructuring can be noticed as one of the difficulties for organizations to implement the microservice architecture.

Security consideration for both architectures, challenges in microservice architecture and future of microservice architecture are considered in transition.

A. Security Considerations

There are some security perspectives for both monolithic and microservice architecture. Table 1 describes the security considerations based on the four different

perspectives such as security control, security audits and monitoring vulnerabilities, attack surface and communication and network security.

TABLE I. COMPARISON OF SECURITY PERSPECTIVES

Security Perspectives	Monolithic Architecture	Microservice Architecture
Security control	Centralized with unified authentication and authorization mechanism for the entire application	Distributed in which each service validates all the incoming requests independently
Security audits and monitoring vulnerabilities	Straightforward due to the single codebase and easier to track and fix the issues	Challenging due to separated services and difficult to track and monitor all security incidents
Attack surface	Smaller attack surface because of single-unit system in nature	Large Attack Surface because of different modular services in nature
Communication and network security	Components communicate each other internally lead to reduce network-level attacks.	Microservices rely on communication over a network makes more vulnerable to attacks.

B. Challenges in Microservice Architecture

Microservices often use central orchestration service which is managing and controlling the individual services running on each container to communicate and work together seamlessly to deliver a large application within the distributed architecture. However, complexities can be raised in managing independent services and handling cross-service dependencies. Each service requires coordination to all deployments, updating and scaling that do not affect the entire application. So, microservice applications need to access data across different services. Maintaining the data consistency and data integrity while avoiding performance bottlenecks are quite challenging in microservice applications. The distributed nature of microservice complicates the test process. Automated testing strategies should be implemented in both the unit and integration levels. Debugging process is challenging as it is difficult to trace the flow of the requests and identify what error occurs. Therefore, microservice development teams need to be prepared for appropriate strategies such as event sourcing, data replication, and consistency protocols for managing distributed data and enhancing the reliability and consistency of their microservice

C. Future of Microservice Architecture

As microservice architecture is more popular and organization recognizes the benefit of microservices including modularity, scalability, and maintainability, the adoption of microservice architecture grows significantly. It enables faster development and shorter time to market. Therefore, it is more suitable for development in terms of continuous integration (CI) and continuous delivery (CD) of applications to deploy updates and new features more frequently. Container orchestration platforms like Kubernetes also improve service discovery, automated scaling, and self-healing capabilities which help organizations to manage all microservices seamlessly with faster speed. Adoption of API gateway and communication protocols enable to build inter-service communication more efficiently with enhanced security and optimize the service interactions with low latency and improved performance.

V. IMPLEMENTATION

To compare the two architectures, the two sample web applications are developed using the same technology stack called MERN (MongoDB, Express, React, and Node. Js). This technology stack enables development with full Javascript ecosystem which provides speed, flexibility, scalability and smooth development across frontend and backend. NodeJS and ExpressJS are used for backend development, ReactJS is for frontend development, and MongoDB Atlas is for database. Besides, Rest API is used for service communication.

This part of the study involves the creation and deployment of an ecommerce web application to a local machine with specified hardware specifications such as 8GB RAM, 256GB SSD storage, and 1.6GHz CPU with 4 cores. Figure 1 shows the entities using in the web application. It is developed using two architectural approaches: microservices, monolithic architecture.

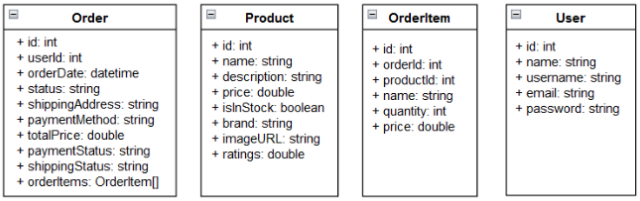

Fig. 1. Order, Product and User Entities for Web Application

The microservices architecture decompose the ecommerce application into smaller, independent services, each responsible for specific functions as Figure 2.

Fig. 2. Client and E-Commerce API Interaction: Users, Products, and Orders

- User Service: The endpoint '/users/profile' handles retrieving and updating user profile information.
- Product Service - The endpoint 'products/id handles for retrieving product details information.
- Order Service - The endpoint 'orders/id handles for retrieving order details information.

Each microservice operates independently, ensuring that issues in one service do not affect others. Front-end components were also designed to interact directly with their respective microservices API endpoints. Figure 3 demonstrates the logical architecture of microservices.

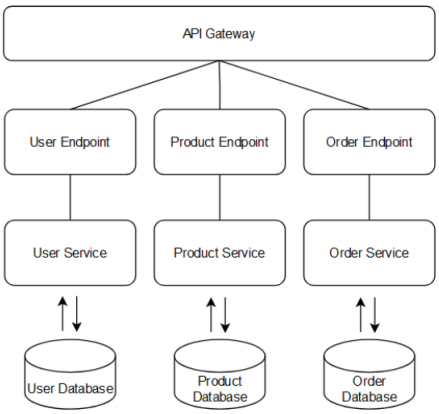

Fig. 3. Microservices Logical Architecture

On the other side, the monolithic architecture combines all functionalities of into a single API for user service, product service, order service, and front-end components as shown in Figure 4. Tests and evaluations are conducted on the local machine assessing for performance, scalability, and resource utilization of both implementations. The advantages and disadvantages of both architectural approaches in web development were obtained through a comparison between microservices and monolithic implementations.

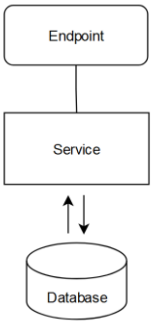

Fig. 4. Monolithic Logical Architecture

A test plan is designed and executed using Apache JMeter to evaluate the performance of both microservices and monolithic architectures. The test plan simulates up to 500 concurrent users, gradually increasing the load over a ramp-up period of 3 minutes. To achieve this, the ramp-up is divided into 10 steps, allowing for a more controlled and steady increase in user activity. Once the target load of 500 users is reached, the system maintains this load for a hold target rate time of 1 minute to assess the system's stability and responsiveness under sustained traffic. This test setup provides valuable insights into the scalability and performance differences between the two architectures.

VI. Evaluation

Concurrency testing, a crucial aspect of performance testing, is utilized to evaluate the performance of both architectural designs. By executing above test plan and gradually increasing the number of concurrent users, the resulting outcomes in terms of throughput, response time and error rate for each architecture can be observed and measured.

A. Response Times

As shown in Figure 5, the test results confirm that microservice architecture give faster response times for different load levels. As the number of virtual users is in increase, the microservice architecture is seen to be a bit faster compared to the monolithic architecture. With 200 concurrent users, the microservices responds in 777 milliseconds which is much faster than the monolithic architecture's 1488 milliseconds. This means that the user traffic increases, microservices' scalability and efficiency gave better performance. Also, the load balancing capability of the microservice architecture is an additional pillar that appears to support its resilience and efficiency, which keeps the performance constant across all virtual user volumes, providing more consistency to the system performance. On primary, it is microservices design that guarantee scalability and performance optimization, particularly on case when workload is higher.

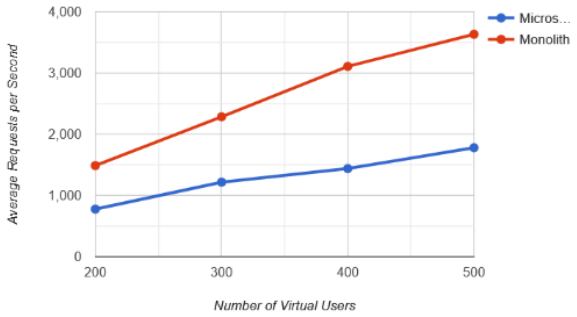

Fig. 5. Comparison of Response Time

B. Error Rate

Another test results obtained from concurrency testing was error rate. In terms of error rates, the microservice architecture constantly showed lower error rates at various levels of concurrency. According to Figure 6, the microservice architecture had a much lower error rate of 19.93% than the monolithic architecture (40.33%) at 200 concurrent users. As the concurrent users increase, microservices constantly perform requests with lower error rates, showcasing fault tolerance and reliability in handling concurrent requests which make them a more suitable choice for scalable and resilient applications.

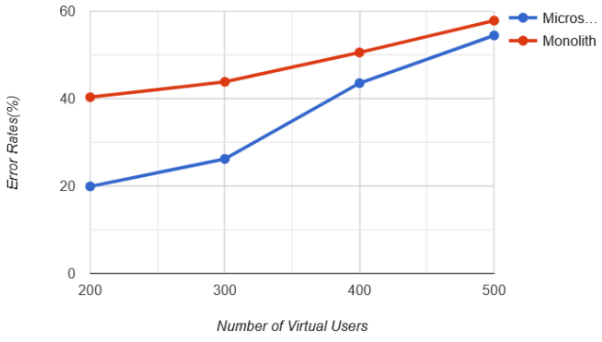

Fig. 6. Error Rate Comparison

C. Throughput

In terms of throughput, it is demonstrated that both the monolithic and microservice architectures exhibit identical levels of performance at this point, with the microservice architecture slightly outperforming the monolithic and exhibiting higher and more steady throughput numbers. At 200 concurrent users, the microservice architecture

performed similarly to the monolithic architecture, achieve a throughput of 161.9 req/seconds, slightly higher than the throughput of 160.4 req/seconds as in Figure 7. Concurrency continued to play a role in this phenomenon, with the microservice pattern outperforming the monolithic pattern in terms of the number of requests handled per unit of time.

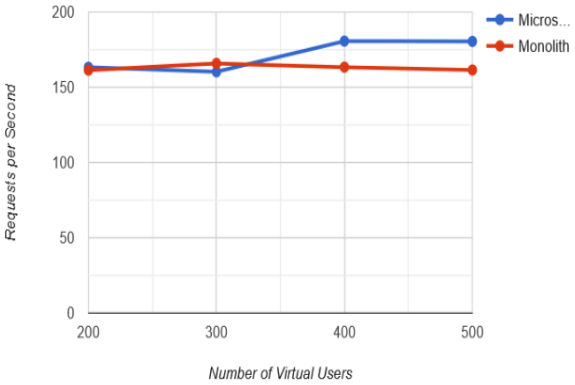

Fig. 7. Throughput Comparison

For example, microservice architecture gave throughput of 180.5 req/seconds at a 400 concurrent users did, which was better than a 158.3 req/seconds throughput for the monolithic architecture. Accordingly, around the 500 concurrent users, microservices transmit a throughput of 178.2req/sec while the monolith give 152.4 req/sec. According to the concurrency test results, microservices give better scalability and efficiency in handling increasing concurrent workloads.

VII. Conclusion

In conclusion, this paper has discussed an analysis of the comparative benefits of microservices and monolithic architectures on web development. On a local machine through implementation, microservices prove to be scalable, flexible and cost effective as monolithic architectures are simple to use in small projects. The load testing comparison between microservices and monolithic architectures reveals that microservices exhibit better scalability and slightly improves performance under increasing virtual user loads. While both architectures handle requests efficiently, microservices show advantages in maintaining consistent performance across varying workloads. These findings suggest a bias towards microservice architecture due to its scalability and responsiveness, making it a promising choice for modern web applications with dynamic workload demands.

References

[1] V. Singh and S. K. Peddoju, “Container-based microservice architecture for cloud applications,” International Conference on Computing, Communication and Automation (ICCCA), 2017, Pages 847–852

[2] M. Fowler, J. Lewis, J. Fowler, “Microservices: a definition of this new architectural term”

[3] J. Lewis, M. Fowler, “Microservices: Decomposing Applications for Deployability and Scalability”

[4] O’Reilly, “Microservices Adoption in 2018: A Survey of O’Reilly Radar Readers”

[5] A. Akbulut, H. Perros, “Performance Analysis of Microservice Design Patterns”, 2019, IEEE Internet Computing, 23, Pages 19-27

[6] S. Salii, J. Ajdari, X. Zenuni, “Migrating to a microservice architecture: benefits and challenges”. 46th MIPRO ICT and Electronics Convention (MIPRO), 2023, Pages1670-1677