A Comparative Analysis of Monolithic and Microservices Architecture

Discipline: Computer Science ***Liji Thomas P.**

***Renjith Roy**

Abstract-In the ever-evolving software development landscape, the choice between monolithic and microservices architectures holds significant implications for project success. This paper aims to compare the monolithic and microservices architectural approaches in software development, intending to assist readers in making informed decisions about which one to use. It explores each approach's strengths and potential challenges, aiming to provide readers with a comprehensive overview. The goal is to help readers select the appropriate architectural framework for building software that aligns with their business needs and requirements. In summary, we will examine how well monolithic and microservices systems can handle growth, remain easy to maintain and facilitate teamwork. We will also discuss a few challenges and drawbacks of both. This information assists individuals in choosing a suitable system for their project requirements and real-world situations.

Keywords—monolithic architecture, microservices architecture, comparative study, software architecture

1. INTRODUCTION

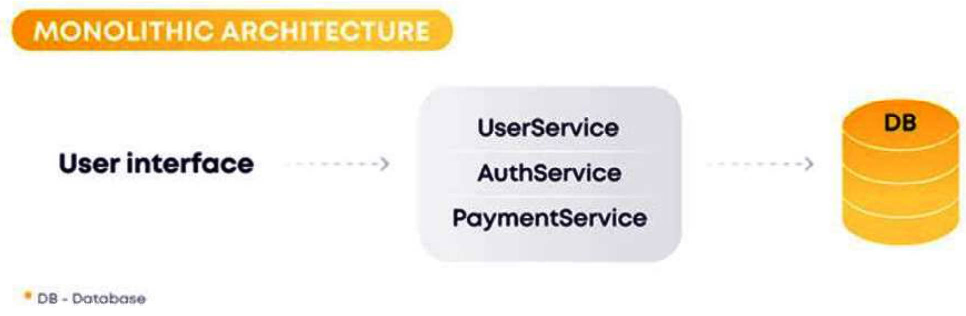
In the world of software, there are two main ways to build things: one is called monolithic, and the other is called microservices. It's like deciding whether to build one big, connected piece or many smaller, independent pieces that still work together. As technology changes, it's crucial to know the strengths and weaknesses of each approach for people who create software. This research takes a deep look at both methods to help understand what works well and what doesn't. The goal is to guide those who build computer programs in making smart choices based on what suits their needs best.

2. MONOLITHIC ARCHITECTURE

[1] Monolithic architecture is a way of designing and building software where the entire application is developed as a single, interconnected unit. In this approach, all the different components and features of the application are tightly integrated and share the same code base. The entire system operates as a unified and cohesive entity, handling all the business processes within a single large network. When changes or updates are needed, modifications must be made to the entire code base, affecting the entire stack simultaneously.

When you think of traditional app development, you usually imply monolithic architecture, when the entire application is built as a single interconnected unit. The monolithic architecture has a single large network with one code base that brings together other components and handles all business flows.

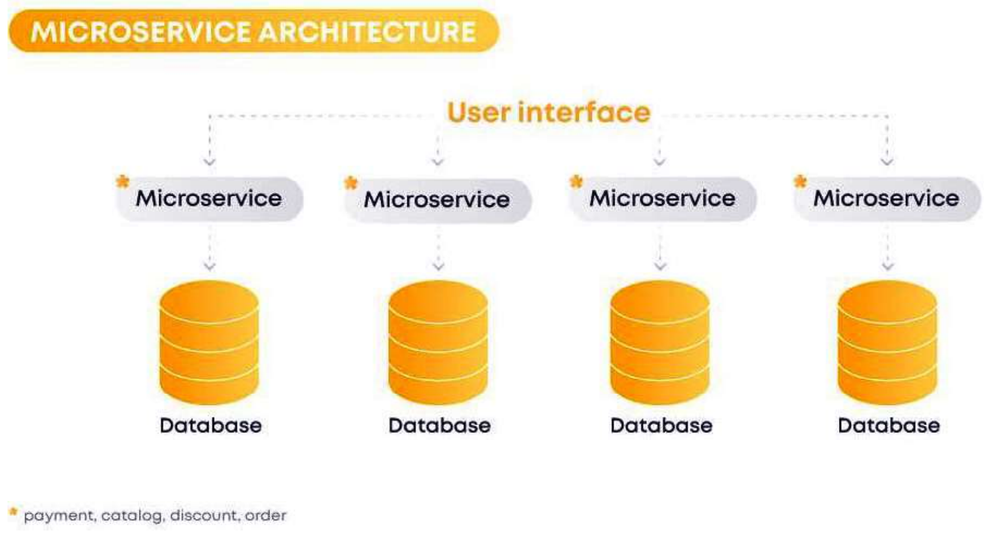
When you want to update or change something, you've got to access the unified code base and make changes in the entire stack at once.



1. MICROSERVICES ARCHITECTURE

[4] A microservice architecture, also simply known as microservices, is a completely different story. Here, you assemble an app from a collection of small, independent services that communicate with each other through APIs or message brokers.

Each microservice is a small application that has its own architecture consisting of a separate business logic and database with a specific goal. So, each service is responsible for a distinct business capability and can be developed, deployed, and scaled independently.



4. PROS AND CONS OF MONOLITHIC AND MICROSERVICES ARCHITECTURE

Exploring the advantages and drawbacks of monolithic and microservices architectures provides crucial insights into the trade-offs associated with each approach in modern software development.

4.1. Monolithic Architecture

4.1.1.1. Pros of Monolithic Architecture

[2] Monolithic architecture has become a traditional solution for a good reason. Usually, it is used at the early stages of a project's life because of the distinctive benefits:

1. **More simple to develop:** It's much easier to develop an app with one code base. Also, it results in a lower development cost.
2. **Easy to test and debug:** As all components are located in the same runtime environment and there is no need to test the interaction between different microservices, testing and debugging are usually simpler and more efficient.[5]
3. **Prompt and simple to deploy:** One executable file or directory significantly simplifies the deployment process.

4. **Good performance:** Monolithic apps may be faster since all the components are in the same runtime environment and there are no network calls involved.

5. **Easy to manage cross-cutting concerns:** Such things as logging, error handling, caching, and performance monitoring are easier to handle because all the code is located in one place.

4.1.1.2. Cons of Monolithic Architecture Monolithic applications can be quite effective until they grow too large. Here are the things that do not make running and maintaining monolithic software any easier:

1. **Scaling:** You won't be able to scale the software components individually, only the entire app. A full application redeploy is required for each update.

2. **Changes are interconnected:** Changes in one part of the application can have some unexpected effects on the entire project.

3. **Performance issues:** If there is an error or bug in one module, it might accumulate over other servers and there might be a risk of shutting down an entire application.

4. **Lack of flexibility:** Changing frameworks or languages will affect the entire app. This means that to implement better technology, you've got to completely rewrite the whole code base.

4.2. Microservices Architecture

4.2.1.1. Pros of Microservices Architecture

[3] DevOps and microservices work together closely. They are like the main building blocks for continuously delivering software, allowing the app to quickly adjust to what users need and what the business aims for. So, microservices bring a bunch of advantages that can make your digital product better, like:

1. **Updating is straightforward:** You can adjust each part independently, simplifying the process of updating each specific component.

2. **Very reliable:** Making changes to one service doesn't impact others, ensuring high reliability.

3. **Troubleshooting bugs is uncomplicated:** Isolating issues in individual services makes it easy to fix them.

4. **Maintenance is easy:** Introducing or removing features can be done without disrupting the entire code base.

5. **Improved agility and flexibility:** This architecture supports agile practices with small teams that deploy changes frequently.

6. **Customizable tech stack:** Your team can select the technology stack for each microservice individually.

4.2.1.2. Cons of Microservices Architecture [5]Microservices aren't a cure-all solution.

They can pose challenges for a project due to:

1. **Complex development and deployment:** Creating more services and placing them in different locations can be complicated.

2. **Management overhead:** Involving multiple teams may increase complexity and make communication more challenging.

3. **Higher cost:** Each service may have its own expenses for testing, deployment, infrastructure hosting, monitoring, and more.

4. **Harder to test:** Testing individual micro services takes more effort, time, and resources.

1. COMPARITIVE ANALYSIS OF MONOLITHIC AND MICROSERVICES ARCHITECTURE

Feature	Monolithic Architecture	Microservice Architecture
Desgin	All components form a single unit	Independent services that can be developed and deployed independently
Development	Easier to develop as all components are built together and share the same codebase	[7]Development can be more complex due to the need to make multiple services work together
Testing	Easier since all components are located in the same environment and interact with each other	More complex as there are multiple services that need to be tested and integrated
Deployment	Simpler as the entire application is deployed as a single unit	Services can be deployed independently, but coordinating their deployment can be more challenging
Maintenance	Can be more difficult to maintain and update as the application grows in complexity	Easier to maintain and update as each service can be updated and deployed independently
Scalability	[6]Scaling up the application requires scaling the entire system, even the components that don't require additional resources	You can scale specific services on demand, allowing for more efficient use of resources
Fault tolerance	A single issue can take down the entire system	You can isolate a specific issue within a separate service to fix it without affecting the entire system
Tech stack	All app components use the same tech stack	Different services can use different tech stacks

6. CONSIDERATION FOR ARCHITECTURAL CHOICE: MONO LITHIC VS MICROSERVICES

6.1 A Case To Stay With Monolithic Architecture

There are several scenarios where a monolithic architecture is a better choice for your project:

1. [8]Monolithic architecture is well-suited for **smaller applications** that have a limited set of functionalities and require less complexity.
2. When you need to **launch a new product fast** (such as an MVP) to validate your business idea or switch from a legacy system to a modern app without delays.
3. If you would like to provide your users with a **fast app with zero to minimal latency**, as there are no network calls involved between different components.
4. You run a **modular monolithic architecture**, which allows you to segment the code into individual feature modules while preserving a unified structure.

6.2 When To Go With Microservices Architecture

These are the cases that better work for microservices:

1. [9]Microservices are a good choice for **large and complex applications** that have multiple functionalities and require a high level of scalability, reliability, and maintainability.
2. As you **plan to roll out new features often**, the microservice architecture allows adding new functionalities separately in each service without a need to redesign the whole software.
3. In case you **want your development team to be flexible** in their flows, microservices is a good fit as with it they can use various tech stacks for different services within an app and work independently on different services.
4. If you **appreciate resilience**, microservices are your choice to go since a single service failure does not affect the entire system.

1. CONCLUSION

In conclusion, the selection between monolithic and microservices architecture is a crucial decision that should be driven by the specific requirements and objectives of your application. It is paramount to carefully assess the functional and non-functional aspects, as well as the anticipated growth and complexity of the software.

[10]The overarching principle is to avoid the pitfalls of overengineering, which can lead to unnecessary complications and hinder the development process. If the nature of your application does not demand the inherent advantages of a microservices architecture, it is advisable not to impose this structure. On the contrary, if there is foresight that your application will evolve into a more intricate system, it would be unwise to confine yourself to a monolithic approach that might struggle to accommodate future complexities.

Balancing practicality and scalability is the key to ensuring the longevity and success of your application. By aligning your architectural choice with the actual needs of your project, you can optimize development efforts, enhance maintainability, and facilitate seamless scalability when required. The decision-making process should be characterized by thoughtful consideration of the trade-offs involved, ultimately aiming for an architecture that best aligns with both the current and future demands of your application. In this dynamic landscape, adaptability and a keen understanding of your application's unique requirements will guide you toward a well-suited and sustainable architectural choice.

REFERENCES

1. O. Al-Debagy and P. Martinek, "A Comparative Review of Microservices and Monolithic Architectures," 2018 IEEE 18th International Symposium on Computational Intelligence and Informatics (CINTI), Budapest, Hungary, 018, pp.000149-000154, doi:10.1109/CINTI.2018.8928192.
2. L. De Lauretis, "From Monolithic Architecture to Microservices Architecture," 2019 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW), Berlin, Germany, 2019, pp. 93- 96, doi: 10.1109/ISSREW.2019.00050.
3. F. Ponce, G. Márquez and H. Astudillo, "Migrating from monolithic architecture to microservices: A Rapid Review," 2019 38th International Conference of the Chilean Computer Science Society (SCCC), Concepcion, Chile, 2019, pp. 1-7, doi: 10.1109/SCCC49216.2019.8966423.

4. G. Blinowski, A. Ojdowska and A. Przyby³ek, “Monolithic vs. Microservice Architecture: A Performance and Scalability Evaluation,” in IEEE Access, vol. 10, pp. 20357-20374, 2022, doi: 10.1109/ACCESS.2022.3152803.
5. K. Gos and W. Zabierowski, “The Comparison of Microservice and Monolithic Architecture,” 2020 IEEE XVIth International Conference on the Perspective Technologies and Methods in MEMS Design (MEMSTECH), Lviv, Ukraine, 2020, pp. 150-153, doi: 10.1109/MEMSTECH49584.2020.9109514.
6. J. -P. Gouigoux and D. Tamzalit, “From Monolith to Microservices: Lessons Learned on an Industrial Migration to a Web Oriented Architecture,” 2017 IEEE International Conference on Software Architecture Workshops (ICSAW), Gothenburg, Sweden, 2017, pp. 62-65,doi:10.1109/ICSAW.2017.35.
7. T. Salah, M. Jamal Zemerly, Chan Yeob Yeun, M. Al- Qutayri and Y. Al-Hammadi, “The evolution of distributed systems towards microservices architecture,” 2016 11th International Conference for Internet Technology and Secured Transactions (ICITST), Barcelona, Spain, 2016, pp. 318-325, doi: 10.1109/ICITST.2016.7856721.
8. G. Mazlami, J. Cito and P. Leitner, “Extraction of Microservices from Monolithic Software Architectures,” *2017 IEEE International Conference on Web Services (ICWS)*, Honolulu, HI, USA, 2017, pp. 524-531, doi: 10.1109/ICWS.2017.61.
9. V. Benavente, L. Yantas, I. Moscol, C. Rodriguez, R. Inquilla and Y. Pomachagua, “Comparative Analysis of Microservices and Monolithic Architecture,” 2022 14th International Conference on Computational Intelligence and Communication Networks (CICN), Al-Khobar, Saudi Arabia, 2022, pp. 177-184, doi: 10.1109/CICN56167.2022.10008275.
10. Y. Abgaz et al., “Decomposition of Monolith Applications Into Microservices Architectures: A Systematic Review,” in IEEE Transactions on Software Engineering, vol. 49, no. 8, pp. 4213-4242, Aug.2023,doi:10.1109/TSE.2023.3287297

***Liji Thomas P.,** Assistant Professor, Dept. of Computer Applications, SCMS Schol of Technology and Management, Muttom, Ernakulam, Kerala, lijithomasp@gmail.com

***Renjith Roy,** Scholar, SCMS Schol of Technology and Management, Muttom, Ernakulam, Kerala, renjithroy06@gmail.com