Design Pattern of Microservices Used in Applications

Nishchita Ramadas Nayak
P.G Student, Department of MCA
RV College of Engineering®, Bangalore, Karnataka, India

Dr. K. S. Jasmine
Associate professor, Department of MCA
RV College of Engineering®, Bangalore, Karnataka, India

Abstract- Microservices is concerning coming up with computer code as a collection of small, loosely coupled services. Recently, microservices application architecture has become a really widespread style in software engineering. It absolutely suits large-scale systems permitting programmers to avoid wasting time on creating changes and lengthening the product. This paper mainly discussed the design pattern of microservices used in some of the applications. A design pattern provides a general reusable resolution for the common issues that occur in software design. The pattern usually shows relationships and interactions between categories or objects. The thought is to hurry up the event method by providing well-tested, verified development/design paradigms. There are design patterns for microservices and those are often divided into 5 Patterns. Every several contains many patterns. Every pattern specifies solutions related to the problem.

Keywords- microservices, design pattern, MSA, SOA, microservice architecture, microservice-based application.

I. INTRODUCTION
Microservices are currently preferred by most businesses over monolithic architectures. Microservices architectures provide enormous opportunity for companies of all sizes, including corporations, to gradually "break" monoliths and adapt their architectures, even if their tech stack is obsolete. There is, however, another side to the coin: not every application adheres to the principles. Microservice-based solutions are presently gaining momentum as a result of they are doing not have the disadvantages of ancient monolithic designs. Business interest in microservices is increasing since the microservice architecture brings a lightweight, independent, reuse-oriented, and quick service preparation approach that minimizes infrastructural risks. Enterprises that have recognized the advantages of microservices have begun to replace their existing package with microservice-based solutions. Visible of this, it's terribly important to accurately verify the application architectures for effective use of this comparatively new technology. The success of microservice architecture depends on the utilization of design patterns. Microservices design patterns enable developers to form separate elements of an application through building an application from a mixture of little services. Every of those services is constructed on an individual basis and deployed separately, in order that they run their processes and communicate with the assistance of lightweight APIs.

II. RELATED WORK
Microservice design patterns are deployed in microservice-based applications. Obtaining evidence on architectural patterns presented in academic and industrial sources, filtering patterns using criteria, and examining their use in microservice-based applications[1], microservice architectures, as well as gaining a thorough understanding of the MSA style and its consequences. The authors discussed the numerous difficulties. The initial difficulty is to identify the key characteristics of microservice architectures, with a particular focus on software features (e.g., security, performance, maintainability). The identification and classification of problems is the second challenge. The third issue is the process of migrating current applications to a microservice-based design[2]. Comprehensive mapping effort to uncover reported microservices usage and derive common patterns and principles based on these use cases. Microservices are gaining traction in the business, with Netflix, Spotify, Amazon, and a slew of large companies adopting them. Several firms are now following suit, moving their systems to microservices. Companies, on the other hand, continue to struggle with selecting the most appropriate architectural patterns, owing to a lack of understanding about the patterns offered. For a set of agreed-upon principles, it can be concluded that distinct design patterns develop for different migration, orchestration, storage, and deployment circumstances[3]. Design patterns for microservices obtained performance statistics for three
microservice design patterns used in the software industry, including query response time, efficient hardware consumption, hosting costs, and packet loss rate. There is no specific microservice pattern that is superior to the others in general. Rather, each design pattern excels under a different set of circumstances. Complex designs necessitate extended development cycles and significant third-party application licence costs[4]. A microservice architecture paradigm that uses a database as a component and is implemented in an application. It also outlines the database's requirements for this pattern and the benefits gained. The suggested pattern has been tested by integrating ebXML registry features into a NoSQL database. Experiments reveal that the suggested streamlined microservice architecture outperforms a SQL-based ebXML registry implemented as a typical Java web service[5].

III. MICROSERVICES DESIGN PATTERN AND ARCHITECTURE
Microservices design patterns enable developers to create discrete components of an application by combining small services into a larger application. Each of these services is written and deployed separately, and they use lightweight APIs to conduct their tasks and interact. The primary purpose of this study is to learn about design patterns and how they are applied to applications. The microservice architecture adheres to a set of general aims or concepts. The four aims to consider in a Microservice Architecture approach are listed below.

✓ Reduce Cost: The overall cost of creating, implementing, and sustaining IT services will be reduced thanks to MSA.
✓ Enable Visibility: MSA support for enhanced service and network visibility.
✓ Increase release speed: MSA will accelerate the time it takes for a service to go from concept to deployment.
✓ Improve Resilience: Our service network will be more resilient thanks to MSA.

Some of the microservice architecture's guiding concepts. They face a number of problems and issues as they work to bring a solution or system to life. For many solutions, these issues are prevalent. These can be overcome by employing the proper and complementary design patterns. Microservices have design patterns, which can be divided into five categories. Many patterns are included within each many. The design patterns available in microservices are shown in Figure 1. These microservices design patterns solved the issues that the apps were having.

Microservices Design Patterns' main benefits
✓ Software engineers can patch bugs and improve a product or its individual components without having to rewrite the entire code — just the parts that relate to a specific feature or service. Software development teams can operate independently because the architecture's services are nearly integrated.

Fig 1. Design Pattern for microservices
Another significant benefit of microservices is their high system availability. The entire product will fail if a piece of the monolith breaks. With MSA, the situation is fundamentally different: even if some services are unavailable due to minor difficulties (such as user authorization), the application will still be usable.

A key feature of the microservices architectural style is the ability to elastically scale the product. It will be necessary for major projects or those aiming to reach a large audience and handle a significant number of inquiries.

When opposed to monolithic design, MSA delivers better maintainability and testability. It's easy to test and manage each module because services are very tiny and loosely connected (changes to one module have no effect on others).

It is easier for a new team member to grasp how the system was designed and how it operates because it is divided into numerous modules, each of which is responsible for a specific task. As a result, the onboarding process is significantly accelerated.

Without collaborating with other software development teams or redeploying the entire programme, each service can be deployed independently. Deployment and debugging processes are simplified as a result.

They have the freedom to choose a different technological stack that will exactly meet the business logic of a specific service, enhancing the system's efficiency.

Figure 2 shows an architecture diagram for microservices based application. The design process begins with several sorts of clients, originating from various devices, attempting to conduct various management functions such as search, develop, and configure, among others. These client requests are subsequently forwarded to identity providers, who authenticate the clients' requests and relay them to API Gateway. The requests are then routed through a well-defined API Gateway to internal services. API Gateway works as an entry point for clients to forward requests to relevant microservices because customers do not call the services directly.

IV. MICROSERVICE BASED APPLICATION

Microservices architecture, or MSA, is a way of developing a single application as a collection of small loosely connected services. Each service is centred on a certain feature or procedure, and it communicates with other services via simple protocols (generally HTTP). MSA is an enhanced version of SOA (service-oriented architecture) that improves the system's modularity, scalability, and adaptability.

What problem does microservices tackle in a monolithic application?

Monolithic applications make it difficult to implement cutting-edge technologies. It's not even easy to go to the cloud.

If one of the system units fails, the monolithic system as a whole will fail. It leads to a lot of downtime.

To scale, the entire system must be rebuilt. Because everything is dependent on and not necessarily compatible with the intended technology to be applied, new technology cannot be simply implemented.
Microservices-based application characteristics:

- Because certain modules are awaiting the conclusion of development on others, development teams must adjust to each other.
- Modules must be developed one after the other, so simultaneous development is not an option for getting to market faster.
- Multiple component services make up an application. The first stage in obtaining service independence is to multiply the services.
- Each service has a set of constrained functional capabilities, known as functional boundaries. It enables dev teams to make minor modifications or even major changes without having to coordinate with other teams.
- Microservices architectures typically include a large number of tools and platforms. The transition from centralised to decentralised systems. The goal of development teams is to create reproducible tools, or technologies that can be utilised by other engineers to tackle common problems.
- The fact that each service inside a microservices-based architecture uses its own database, as opposed to monoliths using a single database across all applications, demonstrates the decentralisation feature.
- Routing that is adaptive. Microservices are distinguished by their transparent routing.
- Cross-functional teams are just a form of cross-functional team. The product is the work of each team, not a distinct component.
- Microservices, as already noted, have a high level of operational stability.

Figure 3 shows microservice based application diagram. In this diagram each service has its own database and is in charge of completing a specific task. One for managing user accounts, one for shipping orders, and one for inventory management. User authorization, payment tracking, transaction processing, and product search are all modules that should be included. One of the most significant advantages of microservices is the lack of dependencies. Instead, the system's components are loosely connected, allowing for easy changes and elastic scaling. If a component of the application fails, it does not affect the rest of the application and it continues to function.

![Microservice based application diagram](image)

**V. CONCLUSION**

While monoliths have long been the standard, microservices have emerged as a viable option to overcome monolithic restrictions. However, this does not rule out the possibility of monoliths becoming obsolete. Just because some people prefer one over the other doesn't indicate it'll be the better choice. It's critical to consider the benefits and drawbacks of each option and to get as much information as possible before making a decision. Keep in mind that a straightforward quantitative evaluation is difficult to make due to the considerable architectural variances. The microservices architecture is revolutionising the way commercial applications are built today. Because it addresses the issue of complexity by
breaking down the application into a set of independent services that are much easier to design, test, and deploy, increasing service availability and overcoming the challenges of monolithic applications. The goal of this study concluded microservices design patterns empower developers to create different frameworks of an application by assembling small services into a larger application. Each of these services is produced and deployed separately, and they use lightweight APIs to execute their tasks more efficiently.

REFERENCES