https://doi.org/10.69624/1816-2126.2024.2.40

Pros and cons of microservices architecture

P.Y. Asadli, R.G.Abaszade, E.A. Khanmamedova
Azerbaijan State Oil and Industry University, Baku, Azerbaijan
parviz.asadli.y@asoiu.edu.az

Abstract: The positive and negative aspects of the microservice architecture were studied in the presented work. Here, in the Software, services are provided in parallel and different from each other. As a result of the stoppage of any service, the uninterrupted operation of other services is ensured. Thanks to our software offering, we are currently identified as the optimal solutions for both users and developers.

Key words: Efficiency, Reliability, Stability, Effectiveness.

1. INTRODUCTİON

In recent years, there has been significant change in the world of software architecture. There is a transition from complex monolithic applications to modular structures consisting of independent services.

The driving forces behind this change are several factors, including managing complexity, improving scalability, and increasing adoption of distributed systems. In this article, we will review the basic concepts of microservices architecture, discuss its advantages and disadvantages, and look at its popular applications.

Microservices are a promising approach to make complex applications more manageable, scalable and flexible [1].

This introductory text provides a basic overview of microservices architecture and encourages the reader to learn more. You can tailor the text to your audience and the focus of your article. One of the important points to pay attention to in the software development process is the process of choosing the right architecture, which has a positive effect on factors such as the speed and optimality of our software. At the same time, it aims to increase the convenience and readability even in the change of employees within the team [2-5].

2. EXPERİMENTAL DETAİLS

Software consists of files with different extensions, which we call codes. At the same time, creating a program is not just about writing code, but also incorporates different skills. During the development and implementation of a software provider, it goes through several stages. Initially, the client's requests and suggestions are obtained by business analysts, then after it is approved, the design process is started, in parallel with this,

technical works are started and reports are given to the client about the work done in the pre-planned time interval. When developing software, we follow certain standards and principles. As a result, following these principles directly affects the quality of our final product. At the same time, the success of a software depends on factors such as its security, stability, and user experience. The software is constantly being improved. Rapid technological developments, changing customer requirements, etc. factors like these accelerate development in software. For these reasons, if the software is completed, the work is not considered complete, and technical support should be provided regularly. Thus, third-party services that are not operational at certain time intervals should be reviewed and if any malfunctions or updates are required, they should be updated. We can also call it preventive care.

As a result, software is one of the integral and important parts of our life today. Software development combines various skills. Despite the difficulties we have listed, it makes our daily work much easier and enables us to do many processes quickly. Software is also a factor in the main source of digitalization. It helps us to perform many operations in our phones and computers that we use in our daily life, from games that we use to tasks that require high logic and calculation. Software is one of the major parts of our day. We can observe this from the online shopping we do on our smartphones to the functionality of our car. Looking at these, we can observe how complex the digital ecosystem is, and it is quite interesting that

it has evolved in parallel with it. As a result, software is an integral part of our world and its development is ongoing. We can look at software as a product made as a result of the joint work of the fields of information technology and the fields and principles of project management.

One of the main considerations when building software is how it will be structured, which has various architectural patterns. If an architect's job is to design a building, the task of a software engineer is to imagine a complex system and adapt it to certain hearts and give the most optimal solution. The difference between them was that the software provider was abstract and flexible. In this article, we'll take a look at the patterns used in software development to ensure its stability, and explore some of the pros and cons. When we look at the product of architectural art, we get aesthetic pleasure and witness a pleasing sight. In software architecture, it is not so much possible to see it with the eye, but we can feel it. So, when we use the software, we can observe it from factors such as the product's operating speed and stability. Just as an architect pays attention to the foundation of a building to make it strong, a software engineer has the responsibility to make the system more stable by following the principles and established hearts of his field. In the construction process, the architect tries to use quality materials, and the programmer engineer is forced to take into account various components at the same time. So, as an example of these, we can show databases, libraries of other vendor companies that are used.

To get a good result, a software engineer must first understand the current situation and accurately analyze the available opportunities so that he can make the right choices. Because these options are the basis of the system.

As a result, software architecture is a unique synthesis of art and engineering. Just as an architect combines aesthetics with engineering principles, a software architect must combine aesthetic vision with engineering realities. This is necessary both to improve the emotional experience of users and to ensure the reliability and operation of the systems. In this article, as we explore this intersection, we will see that software architecture is both an art and a science. When choosing an architecture, we use this way because it is necessary to consider not only the client but also the process of its preparation. So, if we have 5 employees and each of them works in different parts of the project, without a common path, the program will depend on them. However, since we apply any architecture, a newly joined employee will be able to understand the logic by familiarizing himself with the structure.

Recently, there has been a serious innovation in software architectures. In the past, when we built software, we mostly used a monolithic architecture. A monolithic architecture was the way each of our modules was unfragmented within a single application. For example, we roughly divide my software into 2 main parts. It has a view part i.e. frontend part and more functional and backend part for the purpose of establishing connection with the database. Previously, they were built together and it was impossible to imagine them separately. It had its fair share of downsides. So, if any malfunction occurred here, it was relatively difficult to eliminate it because it was not easy to find the error among all the software codes. And in the process of uploading it to the server, it took quite a lot of time because its volume was relatively large. But with microservices, as the name suggests, software is divided into microservices. When this error occurs, we can see as the most important feature that the activity of other unrelated parts is not stopped. In addition, it is convenient to work with a limited small part in the process of maintaining it or adding any new functionality. In this article, we will touch on the positive and negative aspects in more detail. But first of all, I would like to emphasize that although this is a good option, it is not right to use it in every project. Again, an engineer should first analyze the business and then propose a solution. So, if you are building a small project and the number of users you will serve is not large, it is more appropriate to choose a monolithic architecture here. Because in the beginning, you will lose more resources, at least during the creation of the fundamental parts. This will have a negative impact on the budget [2].

Monolithic architecture is an architectural approach that integrates the whole into a single structure. This approach generally refers to a style in which buildings or structures are designed and constructed as a single piece, and where structural integrity and durability are paramount.

We can see that monolithic architecture has been used in many different ways in many cultures and periods throughout history to provide a unique perspective.

For example, in ancient Rome, monolithic architecture was used especially in aqueducts and large public buildings. Roman engineers built buildings from huge stone blocks. These blocks are connected together and greatly compressed, which increased the durability of the structures.

Another interesting example is Rapa Nui (Easter Island), known for its Moai statues and one of the most isolated communities in the world. The Moai statues on this island are monolithically carved from volcanic tuff stone.

Each statue was excavated from an inland quarry and then transported to the shores of the island. The creation and transportation of these statues is a demonstration of the technological and organizational skills of the time.

Today, monolithic architecture is a term commonly used in software development. Here, monolithic architecture refers to a structure in which all software components are integrated and run within a single structure.
This means that parts of the application depend on each other and is an approach often used for large-scale applications.
Compared to traditional monolithic architecture, microservices architecture has many advantages:

2.1 Scalable.

Microservices can scale horizontally as needed. This allows them to easily adapt to growing demands. For example, as the order volume increases in an e-commerce application, the microservices responsible for a certain function can be distributed among more servers.

2.2 Improved flexibility.

Microservices can be easily modified and new features added because they are self-contained. This speeds up the development process and allows businesses to respond more quickly to changing market demands.

2.3 Increased speed.

Microservices can be developed and deployed in parallel. This speeds up the development process and allows for faster rollout of new features.

2.4 Easy Maintenance.

If a microservice fails, other services are not affected. This makes it easier to isolate and troubleshoot the problem.

2.5 Less risk.

Since microservices are small and independent, they prevent errors from spreading throughout the system.

2.6 Other advantages.

Microservices architecture is more suitable for distributed systems, can be more easily integrated with DevOps practices, and is ideal for cloud computing environments [3].
Although microservices architecture is not suitable for every application, it can be used in many different types of applications:
• Large-scale web applications: Large-scale web applications such as e-commerce platforms, social media platforms, and banking systems can benefit from microservices.
• Mobile applications: Mobile applications can use microservices to meet low latency and high availability requirements.
• IoT (Internet of Things) applications: Microservices can be used to process large amounts of data from IoT devices.
Microservices architecture is an increasingly popular approach to software development. It offers many advantages over the traditional approach such as scalability, flexibility, speed and ease of maintenance.
However, microservices architecture also has some disadvantages:
• Increased complexity: Microservices can be more complex than traditional applications.
• Increased Cost: Microservices can increase infrastructure and operational costs.
• Greater skill requirement: Microservices require more skills and expertise than traditional applications.
These drawbacks are also important to consider if you are considering using a microservices architecture [4].

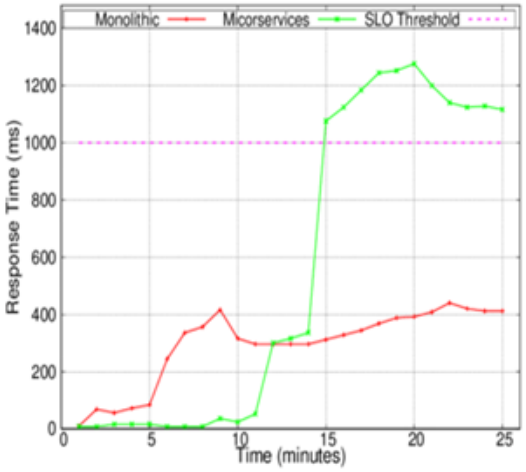

Fig 1. Time diagram.

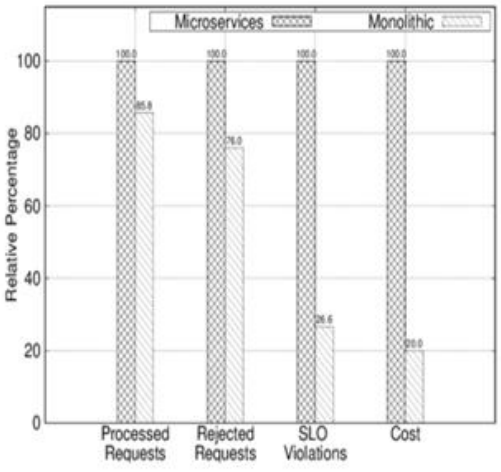

Fig 2. Differences in the diagram.

3. CONCLUSİON

Thanks to this approach, we make our software one of the most optimal solutions for both users and developers. So this gives developers the advantage that they have fewer parts to control when there is a problem. At the same time, downloading them to the server consumes less resources. For the user, if there is a problem with any fast-running service, there is no delay in the work because other services are working.
We can see the differences in the diagram below:

REFERENCES

1. S.Newman, Building microservices, pp.21-24, 2015.
2. M.Kleppman, Designing Data-Intensive Applicatipns, pp.35-40, 2017.
3. L.Bass, P.Clements, R.Kazman, Software architecture in practice, pp.32-38, 1997.
4. E.Solberg, The transition from monolithic architecture to microservice architecture, pp.17-20, 2022.
5. M.Ndungu, Adoption of the microservice architecture, pp.19-21, 2019.