What is the paper about?

Paper describes the 2 main architectures: monolith and microservice. It explains what are those, what are the downfalls/benefits, represent the architecture diagram for each and when and why we should use one approach other than the another.

What is a monolithic application?

Monolithic architecture is seen as the traditional method of application development. An application in a monolithic architecture is developed as a single package. With all the components working hand in hand with each other and is aligned with OOP principles.

What is a microservices-based application?

A microservice architecture includes a set of small, independent and self-contained modules that perform various services. Each service should be able to independently implement the corresponding business units. Each module can be written in any language, has its own database and can communicate with other parts of the system through APIs.

In a paragraph, provide a comparative analysis between the 2 approaches.

The advantage of developing an application that follows the Microservices architecture is that developers have a wide range of possibilities for different technologies such as operating systems, platforms, programming languages to create an application. While in a monolithic architecture, developers are forced to use only one technology, regardless of its limitations.
For example, if your application infrastructure is outdated, moving to a better infrastructure will be very difficult. Highlighting faults in a microservice: even if an error occurs in any of the processes, the rest of the processes will not be affected and can be started, since all services are independent and isolated from each other. But in a monolithic architecture, any kind of misbehavior in any of the components can seriously affect the operation of the entire application. Scaling the application in all microservices are built as different modules. This leads to the division of the team into different jobs, which additionally helps them to easily change and update production. This makes it easier to scale the application. Scaling applications in a monolithic architecture is challenging for developers because it is a cohesive whole. Developers cannot work on individual modules. If possible, heavy coordination will be required during deployment and development.

Why would one want to migrate from one type of application to another?

Monolithic architecture is preferred for developing very small, simple, and lightweight applications. Since monolithic architecture is considered the traditional way of developing applications, it is always best to have a good knowledge of them. Microservice architecture is good for developing complex applications. There might be simpler ways to achieve goals or address the issues that you identify. For

example, if you want to scale your application up faster, you might find that autoscaling is a more efficient solution. If you're finding bugs in production, you can start by implementing unit tests and continuous integration (CI). When you develop a monolithic application, you have to coordinate large teams, which can cause slow software development. When you implement a microservices architecture, you enable small, autonomous teams to work in parallel, which can accelerate your development.

When starting a new project, which approach is best? Why?

There is no “best approach” the best is considered the one that fits the system needs. Not all applications are large enough to break down into microservices. Also, some applications require tight integration between components also the system design and communication is quite time-consuming. Microservices will fit best for applications with big “potential” which will grow gradually. In such system in later stages will be easier to maintain the code, change it and replace the code or stack.

How do these approaches compare in the context of maintainability?

Monolithic applications after 5 years from deployment will be much harder to maintain than the microservice ones. The legacy code will do much trouble, the new versions might be incompatible with previous and amount of code and complexity will just increase with every new modification and addition of new feature. While in microservices it will be easier to update a single separate service and not to take into consideration the other parts of the components, and in case of failure not whole system will fall but just that one service “under maintenance”.