International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 06 Issue: 07 | July - 2022 Impact Factor: 7.185 ISSN: 2582-3930

MICROSERVICE VS MONOLITHIC ARCHITECTURE

Author: Poonam Sawale. Guide: Prof. Shubhangi Mahadik

Bharti Vidyapeeth of Information Management and Information Technology

Abstract:- With the high speed Internet today and the requirement of high speed and reliability in enterprise application world, one has to rely on careful programming choices. It is essential to pick an appropriate software architecture in order to be able to ensure that the app performs well with proper security features. The most common architectures chosen are the Monolithic and Microservice. These are the same because they can achieve the same results with different advantages and disadvantages. In this article, Monolithic and Microservice will be used to mean "the same coexisting functions called by the app whenthe app is run as a monolithic application or microservice application.

Keywords:- Microservice, Monolithic architecture, Microservice architecture, Monolithic.

I. INTRODUCTION

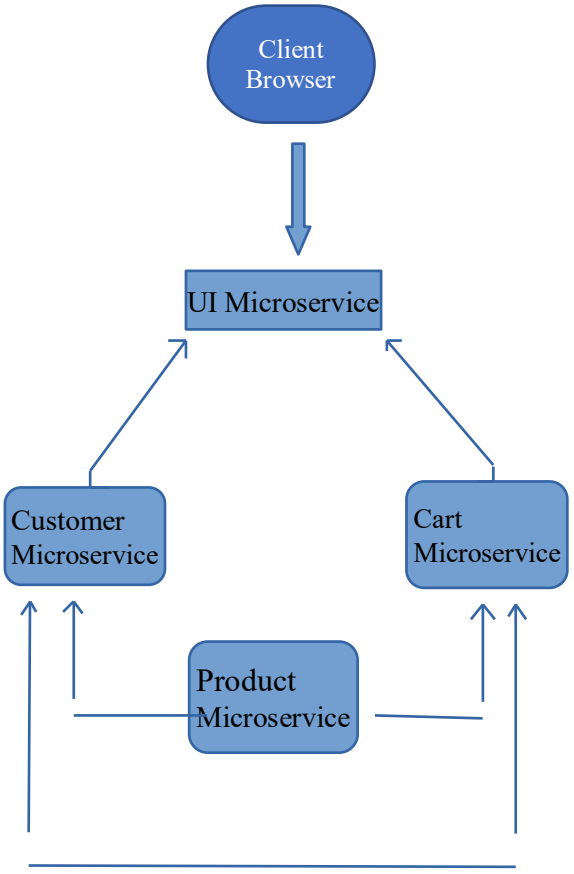
Monolith vs Microservices is the type of a debate that most programmers are faced with in their professional lives. From a high-level view of the issue, monoliths are limiting (for those who have to maintain) and are considered as a real problem from an organizational aspect. However, microservices gives benefits like scalability and flexibility to the solution and could provide a cost-effective solution. But are they really that different or are there othere aspects which can be considered while choosing which application to use?

II. MICROSERVICE:

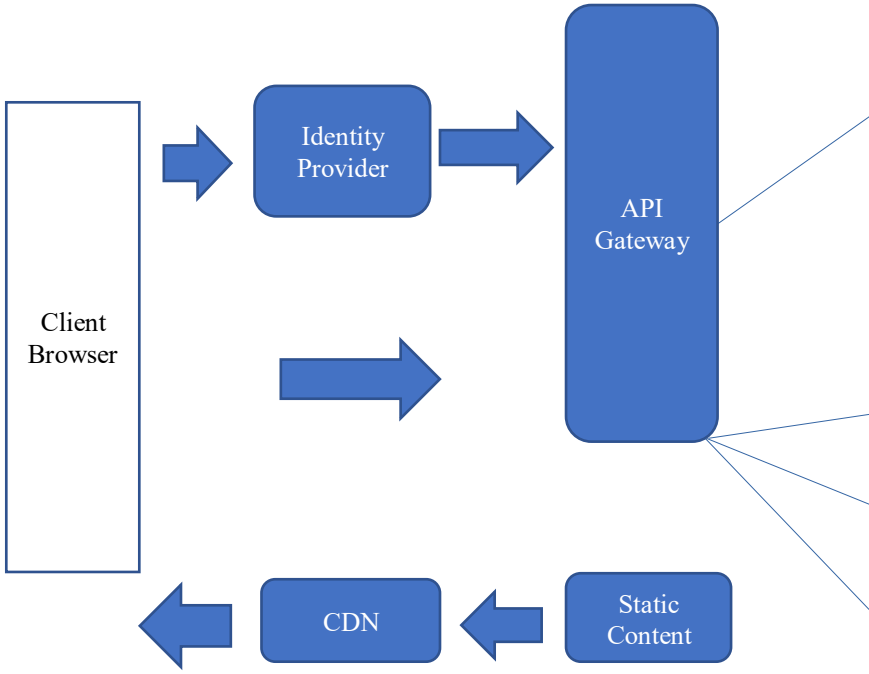
A microservice is a loosely coupled collection of software services embedded into an application. Even though the definition of microservices is somewhat complex, it's simply a software development technique. A microservice consists of a series of small, simple services but their chain forms a complex web.

With microservices, you can build distributed applications using containers. Because each function behaves as an independent service, each function of the application gets its name. As a result, each service can be scaled or updated without affecting other services in the application.

The use of microservices enables IT organizations to become more agile. Scalability is achieved through the use of microservices. As a result, development and change cycles are sped up.



The major advantage of breaking down is – one service can focus on only one capability which will lead to a better quality and it obviously becomes easy.



III. Microservice Architecture

Microservice Architecture is an architectural style that structures an application as a collection of small autonomous services moduled around a Business Domain. In this each service is self maintained and implements a single business capability.

If you have a largs application built using microservice architecture , it can be broken down to small multiple services which together access one large system, but behind the scenes its microservice.

The multiple services present in microservice architecture do not share their data structure but they can communicate through an API.

IV. For Eg. An E-Commerce Site

In Monolithic, all components are divided into single modules like Customer service, Product service and card service., divided into different modules and communicate each other by using well-defined REST or other messaging services. The communication is done through Stateless communication.

Where each pair of request and response is independent transactionand because of this microservicecan communicate effortlessely moreover in microservice architecture the data is federated.

Each microservices response for its own datamodel and data because of which interaction with each microserviceis handled by different instances unlike in monolith where we had only

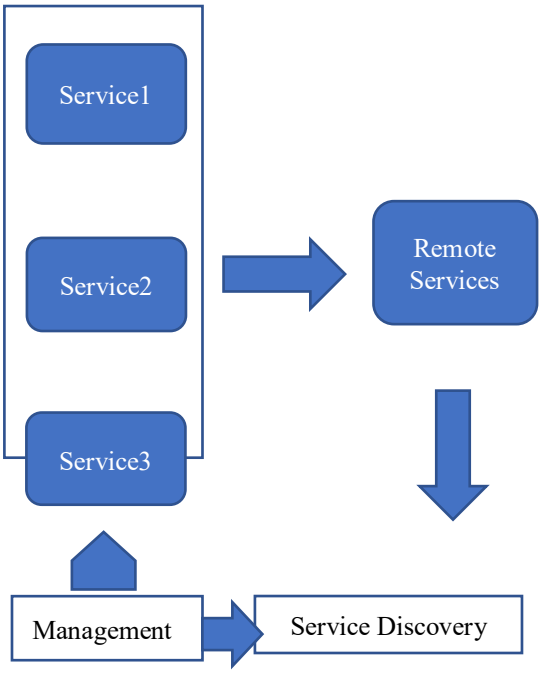
one instance, here we have different instances for different services.

Each microservice has its own data, data model and instance, where services are small, loosely coupled. Each service have small codebase, managed by small development team and each managed independently.

Microservices Architecture comes with quite a few advantages:

1. Independent Development:

Each microservice can be developed independently where single developement team can build and test that service.



2. Independent Deployment:

You can redeploy a service without redeploying an entire application. Bug fixes and feature release are more manageable.

3. Fault Isolation:

If a service goes down, then it will not affect the entire system.

4. Mixed Technology Stack:

Teams can use any technology that they think best fit for the development.

5. Granual Scaling:

Services can be scaled independently, unlike monolithic where we have deploy multiple instances to scale.

There are some disadvantages too:

1. High Complexity**:**

Breaking application in different business domain is not an easy task, when your application is very huge.

2. Consistency**:**

There was a need for our service to give the customer two db, which will have different response elements, at the same time. We had to preserve the consistency of our service by honoring each db response element and the response should be consistent.

3. Automation:

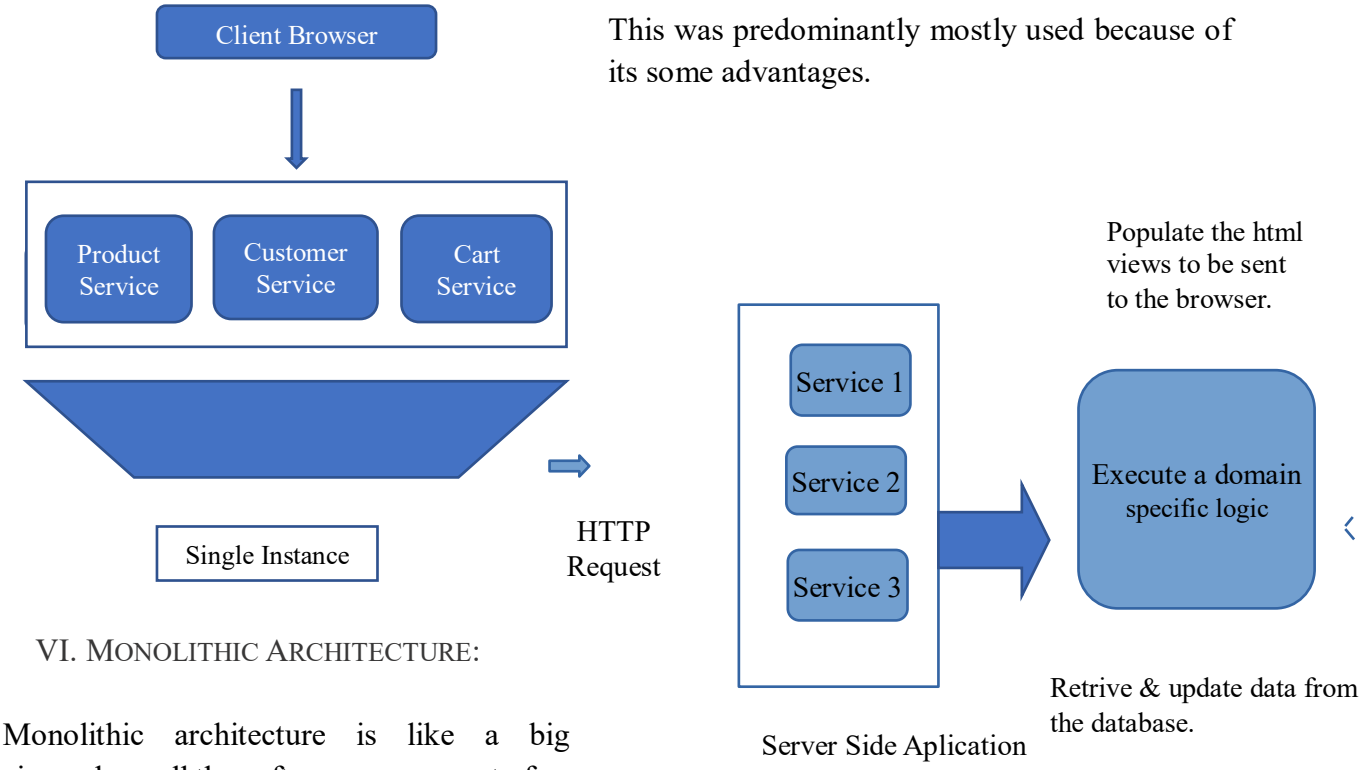
Because so many things happen at once, you need to be quick on your feet if one thing doesn't work correctly. If a service disappears, you need to get that service back online quickly so it doesn't ruin the whole system. To do this, spin off or effectively automate that service.

V. MONOLITHIC:

Mono means one, lithic means stone. And as the name suggests monolithic application is a one giant box or container in which the whole application or system resides. Any request going to that application or machine from a user/user machine via http request, goes directly to that application server. And that application server

has an inderlined DB which interacts and sends back data or request that the user or client has requested for.

Modules within a monolith are tightly coupled together. A monolith is a single binary that contains the business logic and application. In monolithic applications, there are three main components: a database, an interface, and a server-side component.



VI. MONOLITHIC ARCHITECTURE:

Monolithic architecture is like a big container where all the software component of an application are assembled together and tightly packaged.

All these component dependent on each other.

For eg.:- In server side application they execute a specific logic of domain. it will retrieve an renew data from database and at the same time occupy the hypertext markup language view sent to the browser. It is basically big container it is not divided or parted into small small service or not divided into small component.

In E-commerce website like amazon or flip card we visit this website quite often.in this website we have option like customer service, right product and card service which customer can access through their browser and you launch the application it is deployed as a single monolithic application, in this there is only one single instance.so we have customer, right, card service and when you deploy all these service it will be basically a monolithic application.

This was predominantly mostly used because of its some advantages.

1. Easy to Deploy: It is easy to deploy as it only require to get an instance, set up an instance, set DB ang you are good to go.

2. Low Complexity:

Monoliths are simple. They require no fancy infrastructure. With singular servers, your application and database need not be on two different places.

However, as the number of applications grows along with the team, this approach has a number of disadvantages:

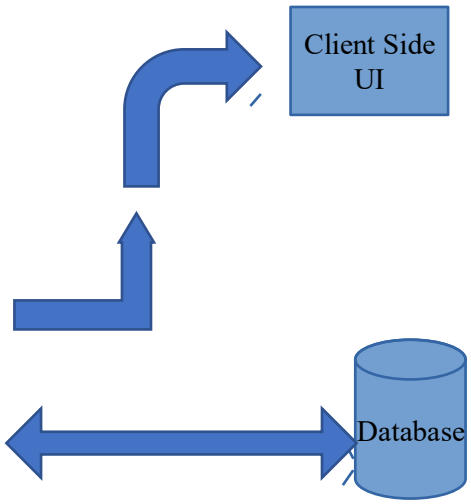
1. Large and complex applications:

In large apps or as the size of apps increases, it becomes really challenging to understand and modify them.

Consequently, development slows down and modularity breaks down, and it becomes more difficult to understand how to correct an error and as a result, code quality suffers.

2. Slow Developemnt:

The application and the respective teams grow to an extent that it becomes difficult to understand and modify, as the app gets increasingly huge in size and there are multiple teams working on it all at once. Also, as codebase grows, testing becomes slower, which means less productivity.



3. Blocks Continuous Deployment:

A largs monolithic application is an obstacle to frequent deployment. For eg. In order to update a component, an entire application needs to be redeployed.

4. Hard to scale:

Each copy of an application will get access to data, which might make caching less effective and increse memory consumptionalong with i/o traffic. Also different application components have different resourses requirement. So with monolithic architecture scaling each components independently becomes diificult.

5. Unreliable:

Because of tightly coupled components, if one component goes down then the entire sysytem will fail.

If there is a bug present in one module then it might affect other modules, as allthe modules are running in one component, which will affect entire application.

6. Inflexible:

Difficult to adapt new framework, languages and applications. It will be extremely expensive in trems of time and cost to rewrite entire application in much better or suitable language or framework.

VII. Which To Choose And When?

If you want to build a prototype quickly, the monolithic design is a better choice for you as executing faster at a time. Because monolithic design offers more flexibility in your applications, with it you can save time in initial stage.

For big complex applications, microservices can be a logical choice because the system is more hacker friendly and modular.

Just like a monolithic building, you could build all of your software on one instruction set (called a monolithic architecture). However, down the road, if you need to split out your code into smaller sets that work well together, you can use microservice architecture.

VIII. CONCLUSION:

Each situation is different. A large problem may require complex logic and the full support of expert services. A particular implementation may be complex and require more programming knowledge. Other factors that should be considered include scalability, time constraints, and technical expertise. A masterful solution may very well require technical complexities that may take time to finish.

To make the most out of the limited duration and budget, you need to consider what products you can offer, and develop something that can create future value.

With regard to simple applications, monolith apps are popular because they are easy to create and deploy. However, if you are creating a small app such as a personal website or a basic web forum, or creating a proof of concept before embarking on a more ambitious venture, monolithic software might be the option.

IX. REFERENCES

1.https://www.digitalocean.com/blog/monolithic-vs-microservice-architecture

2.https://www.divante.com/blog/monolithic-architecture-vs-microservices

3.https://newizze.com/microservices-vs-monolithic-architecture-which-is-right-for-you

4.https://www.geeksforgeeks.org/monolithic-vs-microservices-architecture/

5.https://www.n-ix.com/microservices-vs-monolith-which-architecture-best-choice-your-busine

6.https://www.researchgate.net/publication/341956559_The_Comparison_of_Microservice_and_Monolithic_Architecture