International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 07 Issue: 04 | Apr 2020 www.irjet.net p-ISSN: 2395-0072

Microservices and its Intercommunication using Kafka

V Ganesh Tejas[1], Dr. Hemavathy R[2]

[1]Student, Dept. of Computer Science and Engineering, R V College of Engineering, Bengaluru, India
[2]Assistant Professor, Dept. of Computer Science and Engineering, R V College of Engineering, Bengaluru, India

---

Abstract - *Microservices architecture is a software architecture style where an application is decomposed to a collection of services which are highly decoupled but are capable of communicating with each other. The inter-communication between these decoupled services can be either synchronous or asynchronous. The asynchronous communication can be achieved through Apache Kafka software platform. This paper focuses on the Microservices architecture and the inter-communication between the microservices using Kafka.*

Key Words: **Microservice, Apache Kafka, API (Application Programming Interface).**

1. INTRODUCTION

Microservices architecture is the latest trend in the service oriented architecture where an application is modularized into a number of highly decoupled services, where each service has a distinct functionality and operates independently [5]. All the services are capable of communicating with each other with clear boundaries defined between them. These services work collectively towards accomplishing the main task of the parent application. One of the main benefits of this style of architecture is that each service can be deployed independently, which means that each of the different services can be developed using different programming languages, environments and tools. This makes it possible for continuous delivery of software products and software related services as each microservice is developed independently and is loosely coupled with other microservices. Modifications to any part of the application requires modification only to one of the microservices without disturbing other services. Hence microservices architecture is a light weight, flexible and agile architecture style.

Since all the microservices of an application work towards the common goal, there is a requirement for these services to communicate with each other. For example, the output of one service could be the input for another service, in such cases one service initiates the execution of another service. Such communications are of mainly two types. First is synchronous type of communication such as HTTP (Hyper Text Transfer Protocol) request-response mode of communication, where generally a REST (Representational State Transfer) based API is defined. Second is asynchronous type of communication [4] where we can use software platform like Apache Kafka which acts as an intermediate between the communicating services for asynchronous communication. Kafka creates a pipeline between the communicating services for communication [9]. We shall discuss in detail further.

2. RELATED WORKS

Microservices architecture is a trending software architecture style. Some of the related works follow.

Pooyan Jamshidi has listed out the various benefits of microservices architecture [1], along with the challenges of using this architecture style. Also the various approaches to implement this type of architecture style and the evolution of microservices based architecture are also discussed by him.

Chris Richardson has listed out the benefits of microservices architecture by comparing it with the conventional monolithic style of application development [2]. It is explained that the flexibility which microservices architecture provides is highly benefitting as compared to the rigidity of monolithic style of architecture.

There are many challenges of the microservices style of development which are listed out by Dr. AndreFachatt [3]. One of the main challenges is the complexity in communications between the services due to the dependencies between the services. Also some advantages of this style of design are also discussed.

The asynchronous communication between microservices can be achieved using AMQP (Advanced Message Queuing Protocol) [4]. AMQP is an application layer protocol mainly used for the purpose of asynchronous communication. In this paper we are discussing about Apache Kafka instead.

Microservices based software architecture at the abstract level is to decompose an application to a number of highly decoupled services. But there are many approaches how this can be achieved. The architecture and its various approaches are described in many related works [5] [6] and [7].

Shifting from traditional monolithic architecture to microservices based architecture has its own challenges. But the benefits of using microservices based architecture has an upper hand over the challenges [8].

3. INTRODUCTION TO MICROSERVICES ARCHITECTURE

3.1 Architectural Style

In microservices style of architecture an application is decomposed into a number of loosely coupled services, developed and maintained independently [5]. This can be observed in the Figure 1. These services can be developed in

different programming languages and deployed differently, but all of the services work towards the common goal of accomplishing the parent application task. Microservices of an application are capable of communicating with each other, but have clear boundaries defined between them.

3.2 Benefits of microservices architecture

- It is easier and faster to deploy microservices as each service is developed and deployed independently.
- It is easy to maintain microservices based applications as any issue would be related to any one of the services, we can fix the issue without disturbing other services.
- There is an opportunity to use different programming languages and software platforms to develop different microservices.
- Microservices based architecture gives more flexibility in terms of software release as we can release the services independently.
- Since the application is modularized scaling becomes very easy.

3.3 Challenges of microservices architecture

- The complexity in communication between the services becomes high as the dependencies between the services increase.
- Since each service acts as an independent application, the requirement for resources like storage and band-width is high.
- Testing of microservices based application is difficult due to the disparities between the services.
- Cyclic dependencies arise easily causing problems such as deadlocks.

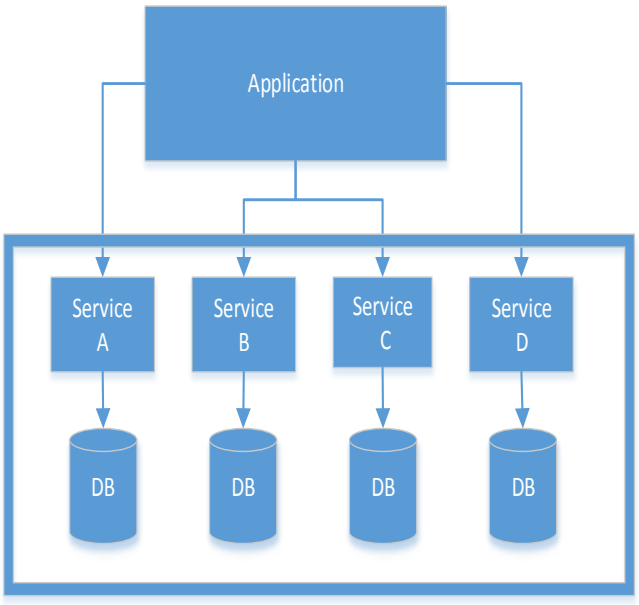

Fig-1: General overview of microservices architecture

4. ASYNCHRONOUS COMMUNICATION

In conventional synchronous mode of communication, we have the request-response type of communication where one service sends a request to another service, which replies by sending the response. Example of such communication is REST (Representational State Transfer) based APIs where request and response are sent through HTTP (Hyper Text Transfer Protocol). Here once the requesting service has sent a request, it has to wait until the response has arrived. This causes blocking of the execution of the requesting service.

To overcome this issue we can use asynchronous mode of communication between the microservices [4]. In asynchronous mode, a service can send another request while awaiting for the response to a previous request. Hence there is no blocking of the execution of the requesting service. Asynchronous communications are mainly triggered by 'events'. So whenever an event is generated, the event triggers the request or response. Hence, there is no blocking of execution of services. Asynchronous communication between services in a microservices based architecture can be achieved using many tools, among which Apache Kafka is one. It acts as an intermediate for communication between services.

5. APACHE KAFKA

5.1 Introduction to Apache Kafka

Apache Kafka is an open source software platform which was designed and developed by LinkedIn. It is now managed by the Apache foundation [9]. It is a stream processing platform which is a type of messaging platform which can handle huge amounts of messages. It is a distributed platform and works based on the 'publish and subscribe' model. In this model there are two main entities, producer and consumer. Producer is the one which produces the message or data and consumer is the one which consumes the message or data. In this model the producer publishes the messages to a data stream, whereas the consumer is subscribed to the data stream. Whenever the producer publishes messages to the data stream, the consumer which is subscribed to the data stream, processes the message or data. This data stream acts as a broker between the producer and consumer for data exchange.

There is no direct connection between the producer and consumer, the message or data exchange between the producer and consumer happens through the broker. When the producer publishes message to the stream, this causes an event that triggers the consumer to consume the data. Hence we can conclude that this is an asynchronous mode of communication as there is no blocking of execution here due to the communication.

5.2 Features of Apache Kafka

- Kafka can handle large amounts of messaging streams easily.
- Kafka is a highly reliable software platform as it can handle failures appropriately.
- Kafka is capable of giving a very high throughput even for large amounts of messages.
- It is a highly durable software platform as it ensures availability always with zero down time.
- Using Kafka we can make transformations to the data coming from the producer by defining new data streams.

6. INTER-SERVICE ASYNCHRONOUS COMMUNICATION USING KAFKA

Intercommunication between the services in a microservices architecture in asynchronous mode can be achieved through Apache Kafka software platform.

In microservices architecture for asynchronous communication between two services, we have the requesting service as our consumer and the service which is requested as the producer. This is mainly because the message from the producer is consumed by the consumer. Now there is no direct interaction between the two services, instead there is a broker in between the two services for message handling, which is Kafka. This can be observed in Figure 2. Whenever two services need to interact, the service which has to send the message pushes the message to the Kafka broker which is called as 'publishing'. The service which needs to receive the message pulls it from the Kafka broker. For this the service has to be subscribed to the Kafka broker. As and when any message is published to the Kafka broker, all the services subscribed to that Kafka broker gets notified. Then services can pull the message from the Kafka broker.

Since the messages or data can be of different types, we can have different Kafka brokers for different types of messages or data. Also within a Kafka broker we can have partitions for different types of messages. Any service pulls only messages of type to which they are subscribed. A service can be subscribed to any number of types of messages in a Kafka broker. In Kafka this message type is called as 'topic'. Each topic refers to a type of message. A producer service can publish messages or data to more than one topic and a consumer service can subscribe to more than one topic based on the requirement.

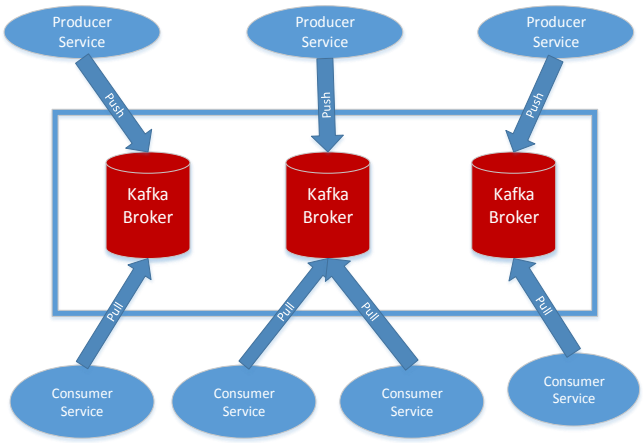

Fig-2: Architecture of microservices with inter-service communication using Kafka

We have three main components of Apache Kafka which are required in our case for intercommunication between services. First we have 'Producer' which is offered as an API that allows a service to publish message or data to one or more topics of Kafka broker. Second, we have the 'Consumer' which is offered as an API that allows the services to subscribe to one or more topics of Kafka broker. Whenever any message or data is published to the topics, the services subscribed to those topics can pull the message or data and process them. Third, we have 'Streams' which is offered as an API that acts as a message or data processor that accepts data from one topic, processes it and sends it to another topic of Kafka broker. This is mainly required for transformation of data between the topics of Kafka broker.

Some of the challenges in this architecture are, since each service is an independent application itself, the requirement for resources like memory, band-width and network is high. Also with the use of Kafka for inter-service communication monitoring the whole application is not easy.

7. CONCLUSION

The major conclusion which can be draw after all the discussions in the previous sections is that, the discussed asynchronous style for intercommunication between services has an edge over the conventional synchronous style. This is because asynchronous style of communication does not block the execution of services, by avoiding the process of waiting for a response and allows for sending one or more requests at a time without having to wait for response. Also the burden of message handling is now removed at the microservice end as the Kafka broker now handles the messages, providing great relief at the microservice end.

REFERENCES

[1] Pooyan Jamshidi, Claus Pahl, Nabor C. Mendonça, James Lewis and Stefan Tilkov, "Microservices: The Journey So Far and Challenges Ahead," IEEE Software(Volume: 35, Issue: 3,May/June 2018) May 2018.

[2] Chris Richardson of Eventuate,Inc.(May 2015), Introduction to Microservices, viewed on 10 April 2020, <https://www.nginx.com/blog/introduction-to-microservices>

[3] Dr. Andre Fachatt (January 2019), Challenges and benefits of the microservice architectural style,Part1, viewed on 03 April 2020 , <https://developer.ibm.com/technologies/microservices/articles/challenges-and-benefits-of-the-microservice-architectural-style-part-1>

[4] Faisal Masood (Sep 06 2018), viewed on 03 April 2020, <https://dzone.com/articles/asynchronous-communication-between-microservices-u>

[5] Xabier Larrucea, Izaskun Santamaria, Ricardo Colomo-Palacios and Christof Ebert, "Microservices", IEEE Software (Volume: 35 , Issue: 3 , May/June 2018),USA,2018,96-100.

[6] Kapil Bakshi, "Microservices-based software architecture and approaches",2017 IEEE Aerospace Conference,Big Sky MT USA,2018.

[7] Paolo Di Francesco, "Architecting Microservices", 2017 IEEE International Conference on Software Architecture Workshops (ICSAW),Gothenburg,Sweden, April 2017.

[8] Arne Koschel, Irina Astrova and Ieramias Dötterl, "Making the move to microservice architecture", 2017 International Conference on Information Society (i-Society),Dublin,Ireland, July 2017.

[9] John Hammink (Feb 2019), An Introdcution to Apache Kafka, viewed on 03 April 2020, < https://dzone.com/articles/an-introduction-to-apache-kafka>