International Journal of Innovative Research in Computer Science & Technology (IJIRCST)
ISSN: 2347-5552, Volume-7, Issue-3, May-2019
DOI: 10.21276/ijircst.2019.7.3.7

A Better Solution Towards Microservices Communication In Web Application: A Survey

Gurudat K S, Prof. Padmashree T

Abstract— **Most of the software applications are configured as client-server fashion in the network in order to enable communication among them. But sometimes these applications must be able to communicate or exchange data between clients and server that are not always up and active. One of the solutions to such issues is to enable applications to store information and communicate asynchronously with other. Thus in this regard Message Oriented Middleware (MOM) which support massage queue based communications are becoming more and more popular today. It is the task of developer to choose the suitable broker for his applications. Thus one has to compare different MOM based brokers to select suitable and efficient ones. This paper mainly compares two brokers-RabbitMQ and RESTful API. By conducting survey on different experimental results it can be concluded, that when large number of users are involved in network communication, RabbitMQ is more stable and suitable to use.**

Keywords*— *RabbitMQ; RESTful API; Message Oriented Middleware (MOM)

I. INTRODUCTION

Microservices is a variant of Service-Oriented Architecture (SOA) that outlines applications as combination of loosely connected devices. This architecture decomposes applications into different working modules and helps developer to clearly understand working of each module and, design and implement every single module with ease. Thus it increases or improves modularity. There are different architectural styles like Representational State Transfer (REST) and Advanced Message Queuing Protocol (AMQP). Representational State Transfer is mainly collection of architectural principles and constraints. RESTful API is an Application Programming Interface that makes task of requesting the resources very intuitive and convenient thus in turn minimizing the complexity of the service provision[1][2]. Most of the scenarios microservices use RESTful API for exchanging messages between clients [6].

Manuscript received May 16, 2019

Gurudat K S, M.Tech Software Engineering, RV College of Engineering®, Bengaluru-59, India, Email:gurudathsbhat@gmail.com

Prof. Padmashree T, Assistant Professor, RV College of Engineering®, Bengaluru-59, India.

But if large numbers of users are involved, sending and requesting messages, then alternative solution like AMQP can be used as RESTful APIs are not so stable in this case. Thus this paper mainly analyses the usage of AMQP as alternative, and discusses different cases of using it. The AMQP is comparatively stable in case of message transfer services. Advanced Message Queuing Protocol (AMQP) is an application level standard that provides integrated messaging services. When large number of users are requesting and sending requests at the same point of time, even if they cannot be processes within the specified time period they can be stored in message queue and can be processed later. The requests are fetched from this queue during the processing time. This paper conducts survey on two main microservices namely-AMQP based RabbitMQ and RESTful API. Performance is analyzed based on number of users and modes of communication. The rest of the article is organized as follows: Section II presents related works in support of survey. Section III describes microservice architectures. Section IV gives insight on analysis of the results from various research papers. The article is concluded in Section V.

II. RELATED WORK

A. Representational State Transfer (RESTful) Application Programming Interface

Representational State Transfer ful (RESTful) Application Programming Interface is one of the architectural system of distributed hyperdermia systems[3]. It is first proposed in 2000 by Roy Rleding. In this architecture the key data abstraction rests in resources. Any kind of information stored can be named as resource. Like images, documents, services, any non virtual objects those are included in tasks etc. This API uses resource identifiers that are used to represent resources, for interaction between components. The state of these resources at any point of interaction time is known as resource representation. The representation consists of several components-like data, description of data i.e. metadata, links that allows users to switch between two different states of resources. The main usage of this API is it provides required flexibilities[6]. The information exchanged does not have dependency on any resources or methods. Thus calling and receiving of different data formats can be easily handled. Because of these advantages RESTful API is used in HTTP environments. It uses GET, DELETE, POST, PUT and other methods to make changes

in any resources. Manipulation of these resources makes actions like calling resources more convenient and flexible. Together with these codes for handling the resources is also made precise and concise. Although this method is flexible and concise, and highly suitable for microservices, it is not applicable for each and every scenario. It is based on HTTP environment, and HTTP will crash if it receives lots of requests, exceeding its capacity. Thus there is need for alternatives like Message Oriented Middleware that can handle large number of resource requests.

B. Advanced Message Queuing Protocol (AMQP)

Advanced Message Queuing Protocol is the standard used in application level for enabling communication between applications or business organizations.[4] It regulates and standardizes the behavior of different service providers for messaging ranging from small services to implementation level, enhancing interoperability of the applications. AMQP is different service, which is different from other services as it allows exclusively specifying the messages that can be received and also allows to trade off reliability, security and performances[5]. The organization or a system that integrates AMQP services have better performances compared to other services. Among other competing paradigms there are several reasons for choosing AMQP services. It is more flexible, allows to connect or integrate the applications on different platforms and it is more convenient to implement and use. Figure 1 shows different functional module of AMQP architecture. It shows functional chain from processing to completion of intended tasks. The message received from application is first received by application module and sent to message queue based on certain specified restrictions. The message queue holds the messages until it is completely processed by the user. Binding module defines interrelation between exchange and message queue modules by providing routing rules.

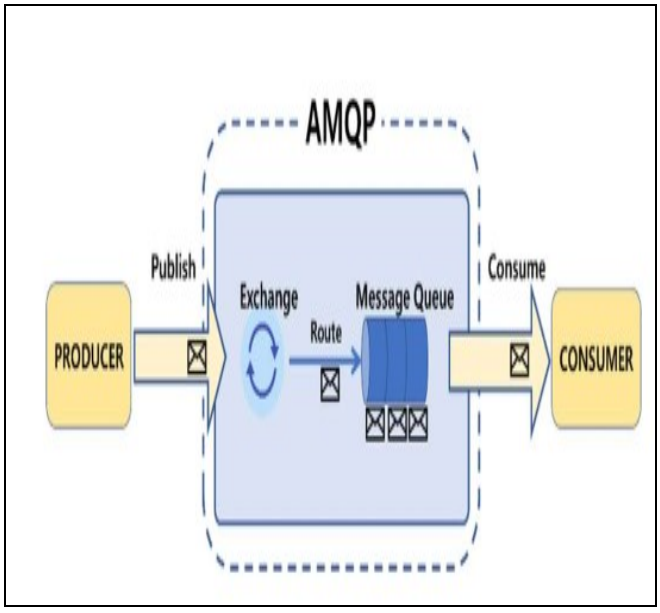

Fig.1: AMQP Architecture

The Figure 1 shows architectural diagram for AMQP service. This paper gives survey on performance analysis by reviewing researches conducted in different papers. RabbitMQ is tested on web applications to allow communication between different applications [1][2]. The performance results are analyzed.

III. MICROSERVICE WEB APPLICATIONS

A. Representational State Transfer API Method

The previous section contains introduction to RESTful API. As already mentioned RESTful API is resource independent. It is also flexible and convenient to use. Figure 2 shows the integration of RESTful API method in the development of web application using microservice architecture. All the interfaces in the application contains RESTful APIs that specifies the actions in advance and implemented on the basis of received URLs and METHODs. Since these actions are defined in advance there is no need of understanding internal structures. All services communicate with each other to process the requests [1].

But RESTful APIs has following disadvantages.

- This API is light-weight, but it cannot handle the complex communication environments.
- It is based on HTTP methods which are not suitable for processing large amount of data.
- Different clients or services are required to test the API services. Thus lot of time is spent to understand their functionalities.
- They also fail to maintain state within the sessions.

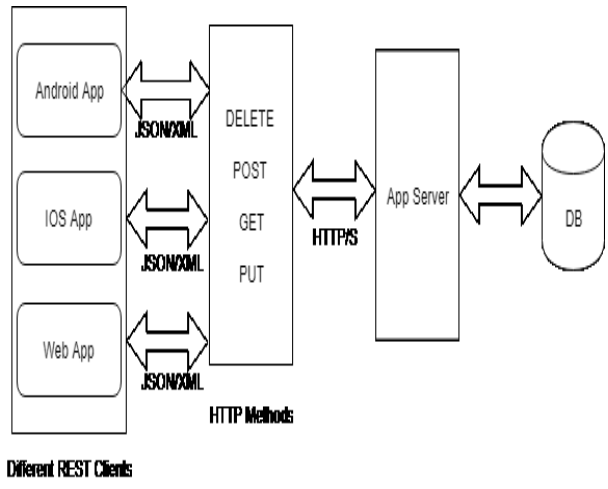

Fig.2: Application using REST API

Figure 2 shows the application using REST API. Here the various types of applications like android, IOS and web app sends REST requests through android device, IOS operating device and browser respectively. The requests are sent to specified service providers using REST API [6]. Later the requests are processed and application server takes specified actions.

International Journal of Innovative Research in Computer Science & Technology (IJIRCST)
ISSN: 2347-5552, Volume-7, Issue-3, May-2019
DOI: 10.21276/ijircst.2019.7.3.7

B. RabbitMQ Method

As the paper proposes comparison between two communication methods used in microservices, the same application is also developed by researchers using RabbitMQ method. This is shown in Figure 3.

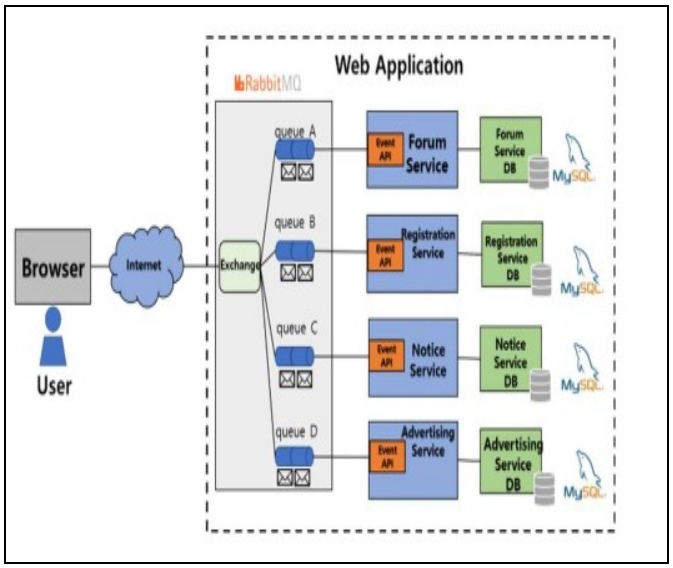

Fig.3: Microservice application using RabbitMQ

Figure 3 shows deployment of RabbitMQ in communication module. The user sends service requests. Here before sending the requests to specified users, the requests are queued to message queues. The message exchange is the core part of RabbitMQ. There are mainly four categories of exchange types- Direct Exchange, Topic Exchange, Fanout Exchange and Headers Exchange. Each exchange type has different overheads on CPU. Thus the developer has to select the suitable one for his applications. The microservice developed using RabbitMQ as shown in the figure requires the messages to be directly sent to the specific application and received by the specific application. Thus Direct Exchange type is used by the researchers [1]. Here after the receiving the request by the exchange module, it is sent to message queue. The exchange module constantly keeps listening the message queue and executes the event as mentioned in the message.

IV. EXPERIMENT AND RESULTS

In the experiment conducted by the researchers, performance of the two microservice based web applications are compared. They have deployed them in various network configurations. The experiments are repeated and average of the result is calculated in order to ensure the accuracy of the experiment conducted. The experiment is conducted on standard configuration using i7 Intel core processor, 16GB memory, 1GB bandwidth, RabbitMQ 3.7.3, Apache 2.4.29. RabbitMQ is more stable when large number of users is involved. Researchers conducted experiment by varying number of users, from 50, 100, 150, 200, 250 300 to 350. These users send request for information in 15 minute time period. The results are shown below. The graph shows the variation of speed at which users receive responses for the request sent.

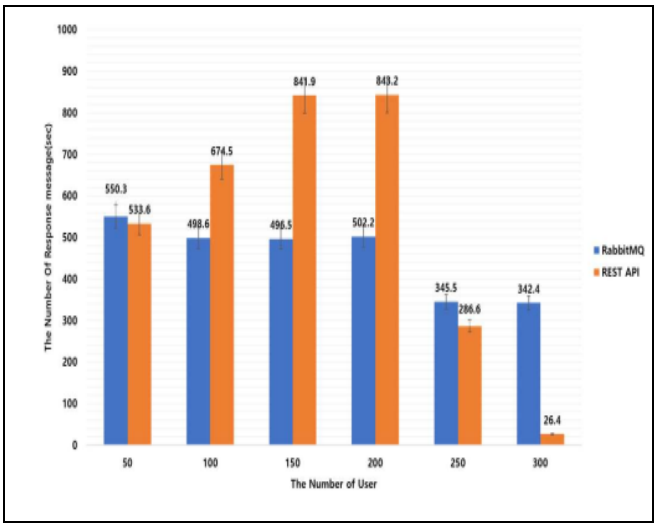

Fig.4: Microservice response speed comparison

Figure 4 shows speed comparison between two microservices using RESTful API and RabbitMQ. The graph shows that if number of users is 50, difference response speed of two microservices is almost negligible. In the initial stages as number of users increases response speed of REST API increases. But as number of users increase from 250 to 300 the response speed of RabbitMQ increases significantly. For 300 users the REST API has highly poor performance in comparison to RabbitMQ.

V. CONCLUSION

The communication methods used by microservice application can contain either RESTful HTTP calls or it can AMQP clients such as RabbitMQ. By analyzing the results it can be concluded that RESTful service can be used as a communication channel between microservices only when there are light traffic in channels. As the traffic in the channel increases RabbbitMQ as a message broker service can be used. Even in small amount of traffic in communication channel RabbitMQ can be proffered as RESTful API uses state transfer method there lies a possibility that when switching from one state to another, some of the packets can be lost. But in message broker services all the messages are stored in different types of queues. Even when the connection between the packets and the channel is lost the data in the queue is retained. When the connection is online the packets can resume their communication. Thus it can be concluded that RabblitMQ a message oriented middleware provides better communication in microservice applications.

ACKNOWLEDGMENT

This survey paper is presented as part of major project for M.Tech, Software Engineering Post Graduate Program in the RV College of Engineering®, Bengaluru-560059.

REFERENCES

[1] Xian Jun Hong, Hyun Sik Yang, Young Han Kim "Performance Analysis of RESTful API and RabbitMQ for Microservice Web Application" IEEE conference, 2018

[2] W. Hasselbring, and G. Steinacker, "Microservice architectures for scalability, agility and reliability in e-commerce," IEEE International Conference on, vol. 1, pp. 243-246, April 2017.

[3] V. Mario, G. Oscar, C. Harold, V. Mauricio, S. Lorena, C. Rubby, and G. Santiago. "Evaluating the monolithic and the microservice architecture pattern to deploy web applications in the cloud," In Computing Colombian Conference, 2015 10th, pp. 583-590, November 2015.

[4] V.M. Ionescu, "The analysis of the performance of RabbitMQ and ActiveMQ," In RoEduNet International Conference-Networking in Education and Research, 2015 14th, pp. 132-137, October 2015.

[5] J.L. Fernandes, I.C. Lopes, J.J. Rodrigues, and S. Ullah. "Performance evaluation of RESTful web services and AMQP protocol," In Ubiquitous and Future Networks, 2013 Fifth International Conference on, pp. 810-815, July 2013.

[6] L. Li, and W. Chou, "Design and describe REST API without violating REST: A Petri net based approach," IEEE International Conference on Web Services, pp. 508-515, July 2011.

