A Comparison between RabbitMQ and REST ful API for Communication between Micro-Services Web Application

Akshay Kamath B[1], Chaitra B H[2]

[1]B.E Student, Department of Computer Science and Engineering, RV College of Engineering, Bangalore
[2]Professor, Department of Computer Science and Engineering, RV College of Engineering, Bangalore

---***---

Abstract - *As Micro-Service architecture is being used as trend technology, it becomes important to discuss what should be the communication medium between multiple micro-services making it efficient and scalable. This paper discusses about why AMQP (Advanced Message Queuing Protocol) technology is better at handling load than traditional REST (Representational State Transfer) communication. AMQP seems to be promising technology specially when there is massive data that is communicated between micro-services. This paper discusses about RabbitMQ as the AMQP Message broker.*

***Key Words*: AMQP, REST, RabbitMQ**

1. INTRODUCTION

RESTful (Representational State Transfer) is a architecture style used to send and receive resource between two piece of software usually they are called as client and server or publisher and subscriber [1][2]. Typically, a Micro-service can use RESTful API to communicate between modules or Send/Receive data, but RESTful API is not reliable when handling large concurrent users thus alternative reliable solution is to use *Advanced Message Queuing Protocol* such as RabbitMQ [1]. This helps in achieving better efficiency and has many advantages, It has a queue where messages are stored and waiting to be processed.

1.1 Representational State Transfer ful Application Programming Interface

Advantage of REST API is that they are very flexible and supports various CURD operation which are Create, Update, Read and Delete. These operations are uses to do all the resource manipulation required and the communication between two or more services [3]. RESTful API used in HTTP environment use GET, POST, PUT, DELETE and some other actions to send and receive resources as shown in Fig-1. Even though RESTful API provide flexibility and all the tools required to communicate between Micro-services, HTTP environment will fail or lose data when the request amount exceeds a limit. Considering requirements like Scalability and Resilience of Software RESTful doesn't seem very promising technology specially for services which has large concurrent users, so a Message-oriented middleware needs to be considered to handle this scenario.

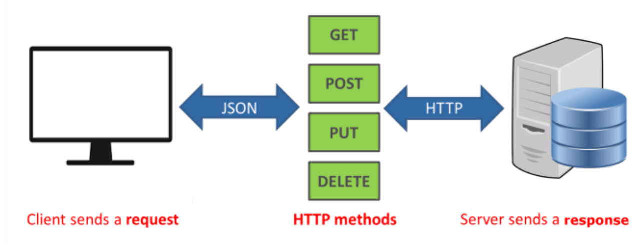

Fig -1: RESTful API operations

1.2 Advanced Message Queuing Protocol

AMQP is advanced Message Queuing protocol, which is quite different compared to RESTful API. This messaging middleware service helps in sending and receiving messages. Message broker is hosted seperately and runs independent of Client or Server Micro-service [4]. This paper discuss about RabbitMQ as the message broker. There are three main components while considering any AMQP messaging service, they are "exchange", "message queue" and "binding". A server sends a message to "exchange" they are then routed to one of the "message queue" they get stored in the queue until they are consumed by the client. Each "message queue" is bound to "exchange" using a binding key, process is called as "binding". "exchange" receives binding key along with message informing "exchange" where to rout the message[5].

System which runs with AMQP performs better, Reason why performance and scalability is highly required for modern systems is competition. The service should be better than competition to create a sense of confidence in the customers and becomes the major part of a business. AMQP technology seems to be promising while considering handling huge data and large concurrent users.

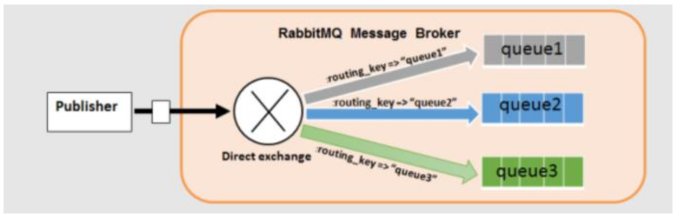

Fig -2: RabbitMQ Exchange and queue [7]

2. Micro-Service web application

In the previous section RESTful API and AMQP is briefly discussed. This section gives a real-world scenario and its implemented using both RESTful API and AMQP in order to learn the design difference and compare them together.

Let's start with a Micro-Service web application that a user can access from browser through internet. User can access database using this API. Various Database operations are supported where user can get or put data on the server through API.

If this application is to be build using RESTful API, there should be an API gateway that contains REST APIs that has actions defined in advanced as shown in Fig-3. They are methods and functionality based on URL. When an API gateway get's the request, it checks the URL and invokes the corresponding REST API and serves the request.

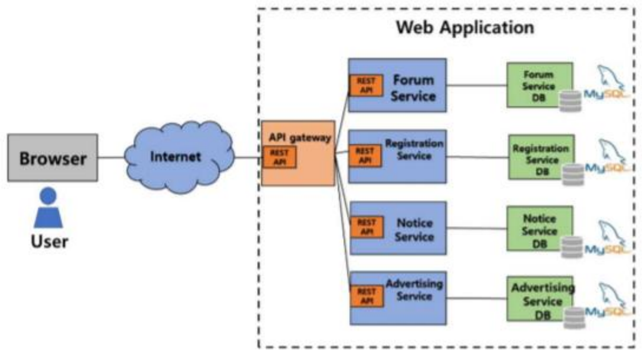

Fig -3: Web Application using RESTful API

If this same application is to be build using a message broker AMQP such as RabbitMQ, API gateway is not required, and REST API is not used here instead we have RabbitMQ information exchange and different queues bound to exchange. Each queue represents a Database Request as shown in Fig-4. Another advantage here is that request gets accumulated and waits in the queue until it is completely served to the user.

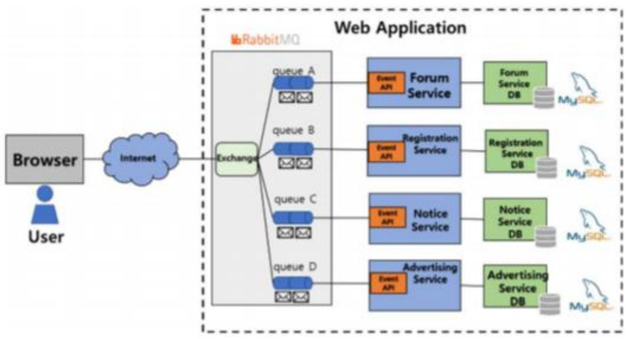

Fig -4: Web Application using RabbitMQ

3. Features of RabbitMQ

Reliability: RabbitMQ is hosted separately and runs independent of the Micro-Service using it. It has Persistent delivery and provides reliability in its functionality.

Flexible routing: RabbitMQ supports routing through exchanges. There are many types of exchanges in RabbitMQ one can use to build the desired routing system. It also supports writing your own exchange through plugin.

High availability: The queue is mirrored in several machines as a cluster. Any failure in Hardware ensures that there is a back-up data and messages are safe.

Platform independent: As said earlier RabbitMQ is hosted separately and Micro-Service using it can be of any language. In most of the cases Publisher and Subscriber Micro-Service are written and deployed in different language.

4. Type of Exchange supported by RabbitMQ

Fanout Exchange: This exchange broadcasts the message to all the queue that it is bonded to irrespective of the key value.

Direct Exchange: Routing key is matched with Binding key and message is routed to all the queue where there is a match.

Topic Exchange: This is same as Direct Exchange where Routing key and Binding key are matched but here Routing key supports wildcard entry.

Header Exchange: This Exchange uses headers to route the message, header is matched with parameters specified in the in-binding queue.

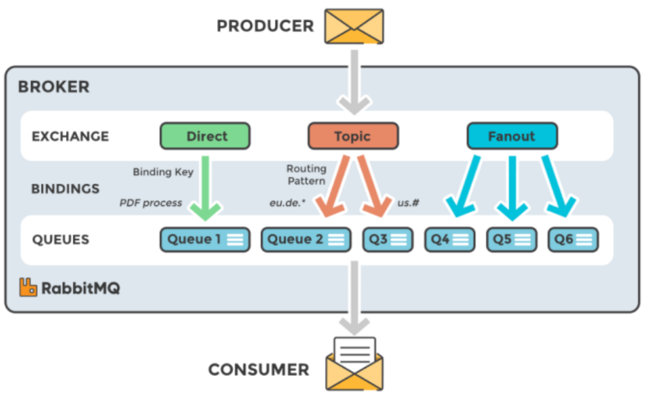

Fig -5: Exchange types [6]

5. Key Advantages of using RabbitMQ over Representational State Transfer

- Message queue provides a Asynchronous communication protocol. Publisher can publish a message to RabbitMQ without worrying if another service is ready to receive it.

- Message will be stored in the queue until the service is not provider, in case the consumer is slower than the producer message waits in the queue.
- The message publishing the service need not know about the working of the service that will process it.
- RabbitMQ will keep the producer and consumer independent of each other.
- RabbitMQ can handle large throughput, especially useful when jobs are longer to run.
- RabbitMQ has a build in user-interface that allows to see all queue and process running visually. The live data traffic entering the data in the queue can be seen.

6. Disadvantage of using Representational State Transfer over RabbitMQ

- REST API is synchronous, and this will cause a dependency on the consuming service unlike AMQP which is asynchronous and there is no dependency on the service consuming.
- Does not perform well when there is lot of concurrent requests coming in. This is the key disadvantage over using AMQP message broker.
- Micro-service that uses RESTful API for communication does not scale very well due to poor performance in heavy load situation.

7. CONCLUSION

After comparing difference between two mode of communication between micro-service and considering advantage and disadvantage of both REST ful API and AMQP, This paper concludes that when there are less number of users it does not matter which mode of the communication medium is uses. When there are less concurrent user requests then both REST ful API and AMQP message-broker performs almost equally but if scalability and reliability is to be considered, AMQP is a better option. Using AMQP will improve the speed and efficiency of the whole micro-service using it.

REFERENCES

[1] 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.

[2] 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.

[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] 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.

[5] 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.

[6] cloudamqp.com/blog/2015-05-18-part1-rabbitmq-for-beginners-what-is-rabbitmq.html

[7] ilearnstack.wordpress.com/2013/04/16/introduction-to-amqp-messaging-with-rabbitmq/