A Review on Enhancement of E-commerce websites performance by using Microservice Architecture

Pranit Mohata[#1], Pritish Tijare[*2]

[1]M.E. Computer Engineering student, Dept. of CSE, Sipna COET, Amravati, Maharashtra, India
[2]Associate Professor, Dept. of CSE, Sipna COET, Amravati, Maharashtra, India

***Abstract*— In today's world, E-commerce websites has huge impact in our day to day life. As there is frequent modification in the website needed to keep them up to date, so its biggest challenge is to keep the site up and avoid any downtime if possible because downtime will lead to loss in the earning and also it disappoints end user. Hence, it's necessary to keep the servers up all the time. With the help of microservices, we can minimize the downtime of Application. Microservices can be deployed, scaled, and tested independently with a single responsibility. They are developed as a small application. They're gaining momentum across industries to facilitate agile delivery mechanisms for service-oriented architecture and to migrate function-oriented legacy architectures toward highly flexible service orientation. In this paper, we will propose implementation of Microservices architecture over monolithic architecture. The proposed Microservices Architecture delivers excellent speed and quality over Monolithic Architecture.**

***Keywords*— Microservices, Monolithic, Webservices, Rest, Microservice Architecture, E-commerce**

I. INTRODUCTION

Microservice architecture - is an architectural style in which application is loosely coupled. The central idea behind Microservice Architecture is that, the application when broken down into smaller, composable pieces or components which work together are easier to build and maintain. Each and every component is continuously developed and separately maintained, and the end application is then simply the sum of all individual components. This is in contrast to a traditional Monolithic architecture style application which are developed all in one piece. In Monolithic Architecture, all the business logic is packaged into a single application process and all the features of the application are written as a separate module that are then packaged into a single main application.

End Applications which is built as a set of individual components are easier to understand and easier to test. The most important advantage of such application is that it is easier to maintain over the life of the application. It helps to achieve much higher agility for the organization and also helps to improve the time it takes to get working improvements to production. When there are large enterprise applications which are developed by teams of geographically and culturally diverse developers, this approach has been proven to be superior.

II. LITERATURE REVIEW AND RELATED WORK

Microservice architectures provide small services that may be deployed and scaled independently of each other, and may employ different middleware stacks for their implementation [1]. Traditionally, information system integration aims at achieving high data coherence among heterogeneous information sources [2], [3]. However, a great challenge with integrated databases is the inherently limited horizontal scalability of transactional database management [4].

One of the intentions of microservice architectures is to overcome the scalability limit of monolithic architectures. A system has a microservice architecture when that system is composed of many collaborating microservices; typically, without centralized control [5]. Microservices are built around business capabilities and did full-stack implementation of software for business area. Automation is key to DevOps success: automated building of systems out of version management repositories; automated execution of unit tests, integration and system tests; automated deployment in test and production environments; including performance benchmarks [6].

The philosophy of the microservices architecture is: "Do one thing and do it well." Services might run within the same process, but they should be independently deployable and easy to replace [7]. Microservice Architectures have the potential to increase the agility of software development [8]. Microservices architectures can be consider as departure from traditional Service Oriented Architecture. Influenced by Domain Driven Design, microservices architectures aim to help business analysts and enterprise architects develop scalable applications that embody flexibility for new functionalities as businesses develop, such as scenarios in the Internet of Things (IoT) domain [9]. Microservices have recently emerged as an architectural style, addressing how to build, manage, and evolve architectures out of small, self-contained units [10].

Villamizar et al. compared the average response time between a monolithic and a microservice-based system in the cloud. In their test, systems were deployed to Amazon Web Services with similar hardware configuration [11]. Villamizar et al. describe their test results as that there is a less performance impact and both systems would fulfil the case study requirements where they had two services, S1 and S2. The case study requirements were defined as: "The service S1 implements CPU intensive algorithms to generate payment plan and their response time is around 3000 milliseconds." and "The response time of service S2 is around 300 milliseconds and this service mainly consumes the database.". Villamizar et al. further explain that hosting their solution would be 17% cheaper for the microservice architecture on Amazon Web Services [11].

Wilhelm Hasselbring along with Guido Steinacker compared the scalability in both monolithic and microservice architecture in one of the companies. They have also explained how developers are now able to do set up, deploy and scale microservices without any support from operations team. With microservices, applications can be scaled dynamically, depending on the current load that a single microservice is facing. Along with that, they have compared the Number of life deployments per week over the last two years and incidents raised because of that. As per the analysis, despite the significant increase of deployments, the number of live incidents remains on a very low level [1].

III. ANALYSIS OF PROBLEM

In Monolithic Architecture, all the business logic is packaged into a single application process and all the features are written as a separate module which are packaged into a single large application. We cannot use different technologies while building monolithic application. For example, if the application is made in Java, then all the components needs to be written in Java and which can be packaged into a war file and then deployed to available server such as tomcat, jboss or a jetty server. Sometimes so many iterations during application development make the application bigger than expected and as we need keep on adding new features, it makes it huge. Once it happens, rapid and effective development and delivery becomes much more difficult. Along with that, bug tracking and fixing also becomes more difficult to handle. Thus, these blocking difficulties result in extravagant time usage. Even for a minor bug fixing one will require to re-deploy whole application. Another big disadvantage is scaling of the application.

If we want to scale, we need to scale the entire monolithic application, even though we will only need more resources on one of the components of the large application. Monolithic application is unreliable, even if single feature of the system caused any issue or does not work, then entire application goes down. Development in monolithic application is quite slow and take lot of time, as we need to build each and every module even after small change in the application. It is not even suitable for complex applications as it makes the application tightly coupled which is not recommended while building large application.

IV. PROPOSED WORK

Number of users who are using E-commerce websites has been increased considerably from last few years. So, there is continuous development and upgradation happens in backend and frontend. This leads to increase in deployment of application for quite a few times which unnecessary increase downtime of application. Our main objective is to reduce the downtime as much as possible by implementing microservice architecture than our legacy monolithic architecture. The proposed project has an approach to software development in which an application is built as a small, independently versioned, and customer-focused services which are scalable and with specific business goals, which will have well defined interfaces and which will communicate with each other over standard protocols. Each service will implement a single business logic and is self-contained.

These services will largely de-coupled, so that application can be easily built, altered and scaled. Microservice architecture support agile development, we can easily develop any new feature if required or even we can discard if not in use. We can have independent deployment of any individual module and these can also be developed independently. This approach makes sure that even if any particular feature of the application failed, other part of the application will keep on working. Thus, making the downtime of the application as much less as possible. As downtime of the application and business profit are directly proportional to each other, if there is more downtime of the application, there are greater chances of loss in business revenue. But if there is no or minimal downtime of the application, it will be profitable for the business. The proposed module will implement microservice architecture and will compare the deployment time required for Monolithic application over Microservice application. The result of the project will be analyzed.

V. BASIC SYSTEM ARCHITECTURE

A. *Basic Monolithic Architecture*

Fig. 1 is the example which describe Monolithic Architecture where all the business logic is packaged into single application. As we can see, there are four different services Login, Products, User and Order service which are doing login into the application, display product list, provide user information and display order status respectively. These are then deployed as a single application either as war or jar.

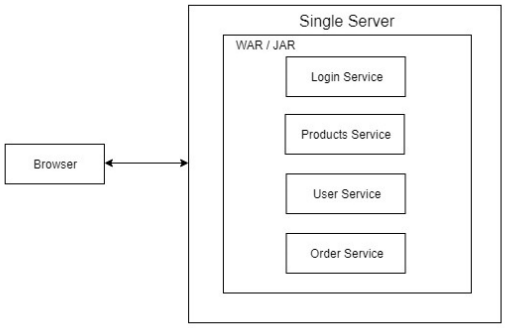

Fig. 1 Basic Monolithic Architecture

B. Basic Microservices Architecture

Fig. 2 is the example which describe Microservice Architecture where each and every module has its own functionality. For Example, Login service will do login related work for the user, User service will give the profile details of the user. These modules are deployed in different server so that if there is any change in any one of the modules, then in that case only module where changes are required need to deploy, not whole application. Thus, it will reduce the downtime of the application. Even if one particular service has lots of hits, then in that case we can scale only that particular module, not whole application.

Also, in future if we need to change the technology of particular module, then this is also possible in Microservice Architecture. In this way we can have multiple technologies that can exist with each other in Microservice Architecture.

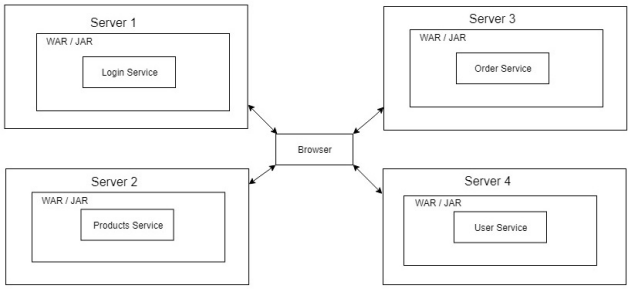

Fig.2 Basic Microservices Architecture

VI. CONCLUSION

In this review paper we are trying to make a system which will help to understand the benefits and advantages of microservices over monolithic architecture by comparing downtime, Complexity and scalability of application.

REFERENCES

[1] Microservice Architectures for Scalability, Agility and Reliability in E-Commerce by Wilhelm Hasselbring and Guido Steinacker

[2] "Information system integration," W. Hasselbring, Communications of the ACM, vol.43, no. 6, pp. 32–36, 2000

[3] IEEE Multimedia, vol. 9, no. 1, pp. 16–25, 2002. "Web Data Integration for E-Commerce Applications,".

[4] M. Abbott and M. Fisher, The Art of Scalability, 2nd ed. Addison- Wesley, 2015

[5] S. Newman, Building Microservices. O'Reilly, 2015.

[6] [6] J. Waller, N. C. Ehmke, and W. Hasselbring, ACM SIGSOFT Softw. Eng. Notes, vol. 40, no. 2, pp. 1–4, Mar. 2015 "Including performance benchmarks into continuous integration to enable DevOps,"

[7] https://dzone.com/articles/benefits-amp-examples-of-microservices-architectur

[8] Microservices: Granularity vs. Performance by Dharmendra Shadija, Mo Rezai and Richard Hill

[9] Dharmendra Shadija, Mo Rezai, and Richard Hill. 2017. Towards an Understanding of Microservices. In 23rd IEEE International Conference on Automation and Computing. IEEE Computer Society, Huddersfield, UK.

[10] Thones. 2015. Microservices. IEEE Software 32, 1 (2015),

[11] M. Villamizar, O. Garcés, H. Castro, M. Verano, L. Salamanca, R. Casallas, and S. Gil. Evaluating monolithic and microservice architecture pattern to deploy web applications in the cloud. In 2015, 10th Computing Colombian Conference (10CCC), pages 583–590, Sept 2015.