Performance Improvement of E-commerce websites using Microservice Architecture over Monolithic Architecture

Pritish Tijare
Associate Professor
Department of Computer Science & Engineering
Sipna College of Engineering & Technology, Amravati, India

Abstract : Microservices Architecture delivers excellent speed and quality over Monolithic Architecture. As requirements from client are ever changing, updating the application will be always there. Updating of the application leads to deployment so that new feature will be updated for the application. So, its biggest challenge is to avoid any downtime as much as possible and to keep the site up. Because downtime causes loss in the earning and also it disappoints the user who uses the application. Hence, servers need to be running all the time. We can minimize the downtime of Application by implementing Microservice Architecture. Microservices works on Single Responsibility Principal and they can be deployed independently. Along with that, they can be scaled and tested independently. They are design and developed as a small module. Microservices are gaining momentum now a days across industries for service-oriented architecture to facilitate agile delivery mechanisms. With the help of Microservice architecture, we are able to migrate function-oriented legacy architectures toward highly flexible service orientation. In this paper, we have implemented Microservices architecture over monolithic architecture which helps in reducing downtime of the application which leads to increase in the performance.

***IndexTerms* - Monolithic, Microservice, Architecure.**

I. INTRODUCTION

A microservices architecture is a collection of small and autonomous services. Each service is self-sufficient and implements a single business capability. Because of this, Microservice Architecture based applications are loosely coupled. The main purpose of implementing microservice architecture is that, when application is broken down into composable and smaller components which when work together will be easier for deploy, build and maintain. In Microservice Architecture, each service has a separate codebase, hence a small development team can also manage it. This is totally opposite to a traditional Monolithic architecture style where end application is developed all in one piece. In Monolithic Architecture, all the features of the application are written as a separate module that are then packaged into a single main application. We should use Microservice Architecture when there is large application and require a high release velocity, also when application is complex and needs to be highly scalable.

In Microservice Architecture, Services are responsible for persisting their own data or external state which differs from the traditional model, where a separate data layer handles data persistence. In Microservice Architecture, services will communicate with other services by using well-defined APIs where internal implementation details of each service are hidden from other services. Another important feature of Microservice Architecture is that, services don't need to share the same frameworks, technology stack or libraries. 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. 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.

II. RELATED WORK

To overcome the scalability limit of monolithic architectures is one of the intentions of Microservice Architectures based application. A system has a microservice architecture when that system is composed of many collaborating microservices; typically, without centralized control [1]. The microservices architecture works on philosophy: “Do one thing and do it well." Services should be independently deployable and easy to replace run even when they are within the same process [2]. Microservice architectures provide small services that may be deployed and scaled independently of each other, and may employ different middleware stacks for their implementation [3]. 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 [3]. Traditionally, information system integration aims at achieving high data coherence among heterogeneous information sources [4], [5]. However, a great challenge with integrated databases is the inherently limited horizontal scalability of transactional database management [6].
Microservices did full-stack implementation of software for business area and are built around business capabilities. 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 [7]. 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 [8]. 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 [8]. Microservices architectures can be consider as departure from traditional Service Oriented Architecture. Influenced by the design which are Domain Driven, microservices architectures aim to help enterprise architects and business analysts to develop scalable applications that embody flexibility for new functionalities as businesses develop, such as scenarios in the Internet of Things (IoT) domain [9].
Microservice Architectures have the potential to increase the agility of software development [10]. Microservices have recently emerged as an architectural style, addressing how to build, manage, and evolve architectures out of small, self-contained units [11].

III. Analysis of problem

A monolithic application is tightly coupled as it is built as a single unit and entangled as the application evolves, thus making it difficult to isolate services for purposes such as code maintainability or independent scaling. A monolith is in short single logical executable due to which even if any alterations to the system is required, a developer needs to build and deploy an updated version of the server-side application. Monolithic architectures are much harder to understand, because there may be dependencies, side-effects, etc. which are not obvious when you're looking at a particular controller or service. 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. Monolithic application is unreliable because entire application will go down even if single feature of the system caused any issue or does not work.
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. 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. 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.

IV. Proposed work

As we know, De-coupled services are easier to reconfigure and recompose to serve the purposes of different applications, Microservice Architecture help us to achieve it. Microservice architectures are better organized, as each microservice follows single responsibility principal means each service has a very specific job, and it's not concerned with the jobs of other components. Thus, making Microservices are loosely coupled. They can also perform better depending on how they're organized because it's possible to isolate frequently called services usually termed as hot services and scale them independent of the rest of the app. These services will largely de-coupled, so that application can be altered, build and scaled easily. Agile development cab be achieved with the help of Microservice architecture, 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 as well as monolithic architecture and will compare the deployment time required for Monolithic application over Microservice application. 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. The result of the project will be analyzed. Our main objective is to reduce the downtime as much as possible by implementing microservice architecture than our legacy monolithic architecture.

V. System Architecture

A. Monolithic Architecture:

Monolithic Architecture can be understood from below figure where all the business logic is packaged into single application. Five different services such as Product, Login, Profile, Order and Payment are present. These services are used for display product list, login into the application, displaying profile information about the logged in user, order details for the user and display different banks available for payment respectively. These are then deployed as a single application either as war or jar.

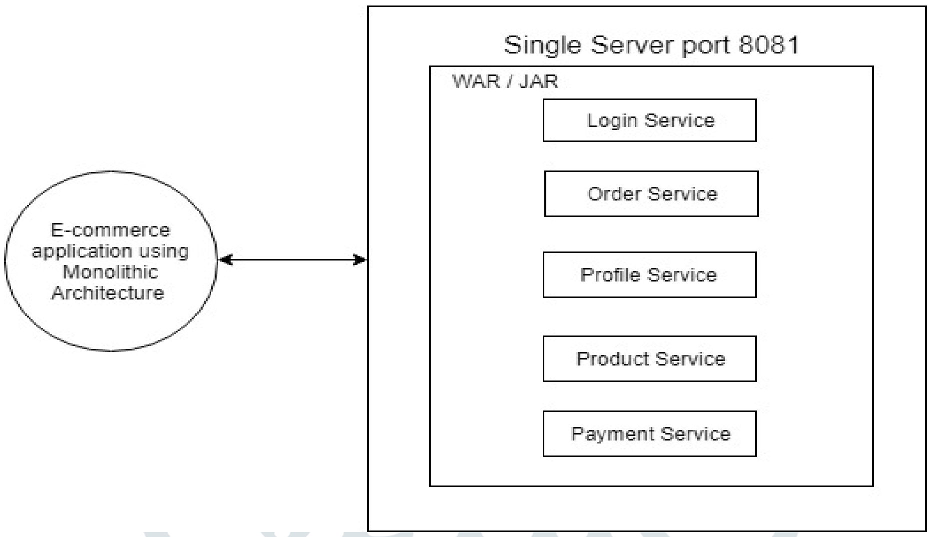

Figure 1: Monolithic Architecture

B. Microservices Architecture:

Microservice Architecture can be understood from below figure where each and every module has its own functionality. For Example, Profile service will give the profile details of the user, Login service will do login related work for the user, These modules are deployed on different server or can be deployed on same server with different port number 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.

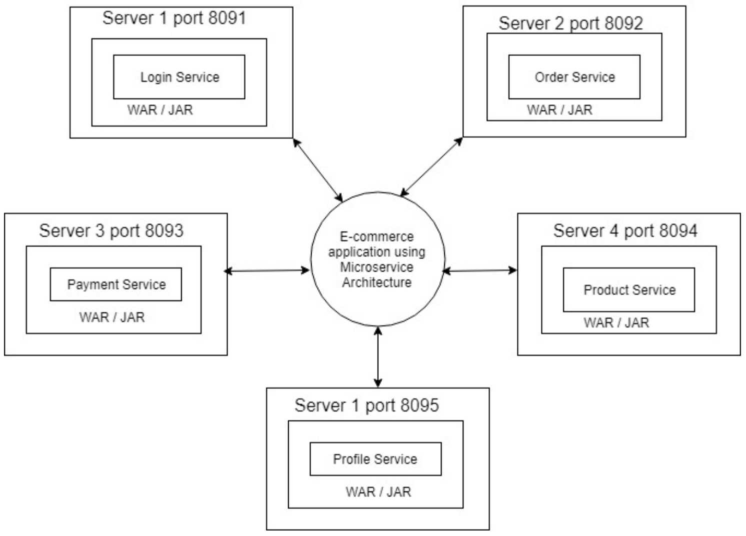

Figure 2: Microservice Architecture

VI. Implementation

We have developed an E-commerce look alike site which has basic services such as Login, Order, Payment, Profile and Product details for the implementation purpose. We have implemented them for monolithic application as well as microservice application.

A. Application implementing Monolithic Architecture

In case of Monolithic architecture, we have all the above-mentioned services deployed on single server. But considering the fact that for any e-commerce site introduction of new functionality on frequent basis is not new, so if client comes up with new functionality, we need to re-deploy whole application even if the changes required are very less. In this case, our application will be down till deployment with new changed is not completed. Hence it creates unnecessary downtime.

For ex. Initially we have an e-commerce platform which has only electronic items. But client now wants their site should have clothing line-up also. In this case, we need to deploy whole application again to introduce the new changes even though changes are only implemented in Product details service. Because of re-deployment, we will get downtime during deployment process. Once deployment is successful, we will be able to login into application and can view both Electronic as well as Clothing section.

Step 1 : We will login into our web application. At present, this site has only Electronic items available for purchasing. It will also give us Profile details and past order details of the logged in user

Step 2: After login, we can see that Electronic Product list is displayed. Along with that, it gives a profile details of the logged in use as well as order history for that user.

Step 3 : There is requirement change from the customer in our current application. Now customer wants to add clothing along with Electronic section. So, after doing changes in our application, we need to deploy it on server, so that users can see both Electronic item as well as clothing items.

Step 4: For deployment purpose, server needs to stop. So, we will stop the server. In case of Monolithic application, our application will be in Maintenance state till the new changes are deployed. Hence, end user will not be able to login into the application

Step 5: Once deployment is completed, user will be able to login into the web application and new changes will be reflected in our application

B. Application implementing Microservice Architecture

In case of Microservice architecture, we have all the above-mentioned services deployed on different server as Microservice architecture follows Single Responsibility Principal. But considering the fact that for any e-commerce site introduction of new functionality on frequent basis is not new, so if client comes up with new functionality, we need to deploy only that service in which changes needs to be incorporated. In this case, only that service in which changes are introduce will be down until deployment is completed, not whole application.

For ex. Initially we have an e-commerce platform which has only electronic items. But client now wants their site should have clothing line-up also. In this case, we need to deploy only product details service again to introduce the new changes. Once deployment is successful, we will be able to see both Electronic as well as Clothing section

Step 1 : We will login into our web application. At present, this site has only Electronic items available for purchasing. It will also give us Profile details and past order details of the logged in user.

Step 2 : After login, we can see that Electronic Product list is displayed. Along with that, it gives a profile details of the logged in use as well as order history for that user.

Step 3: There is requirement change from the customer in our current application. Now customer wants to add clothing along with Electronic section. So, after doing changes in our application, we need to deploy it on server, so that users can see both Electronic item as well as clothing items.

Step 4: For deployment purpose, server needs to stop. So, we will stop the server.

In case of Microservices application, we need to stop only that server where Product details service is called. Hence, we will stop Product service only. In this case, user will be able to login into our web application. He will get Maintenance page on Product list page, but user will be able to see his rest of the details such as Profile and Order history, etc.

Step 5: Once deployment is completed, user will be able to login into the web application and new changes will be reflected in our application

VII. Results & Discussion

We have compared the time required for deployment of both Monolithic and Microservice architecture for each service present on different instances. Its graphical representation can be shown as follow:

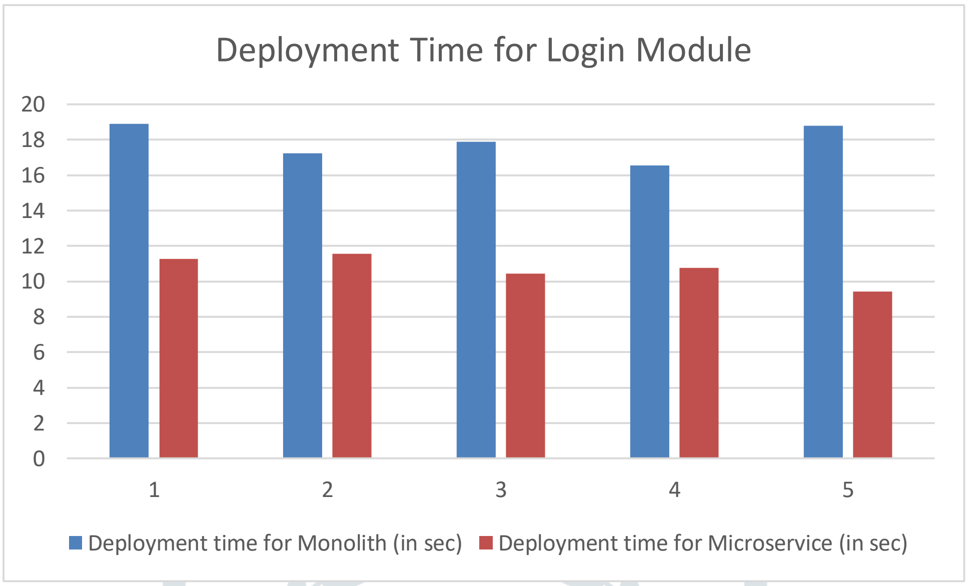

Figure 3: Deployment Time for Login

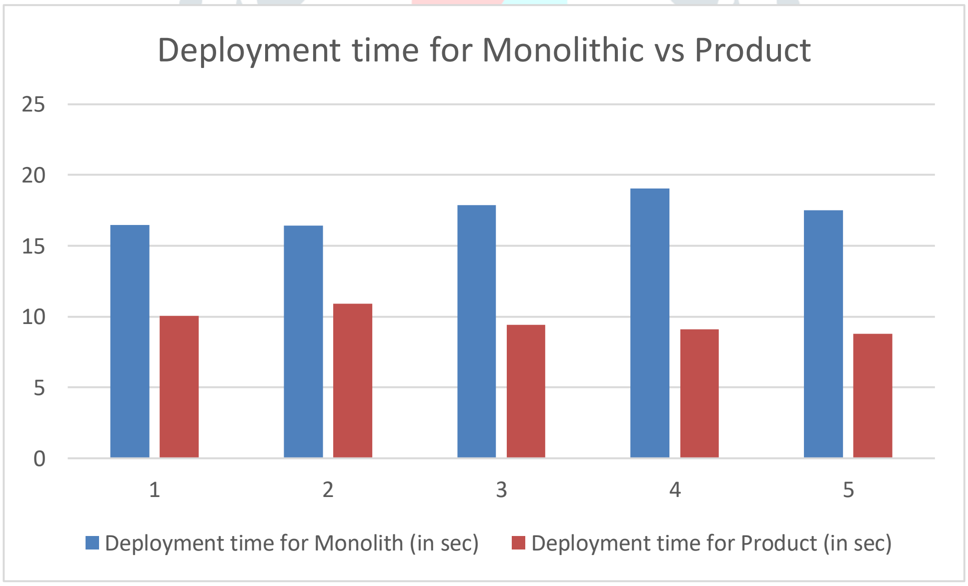

Figure 4: Deployment time for Monolithic vs Product

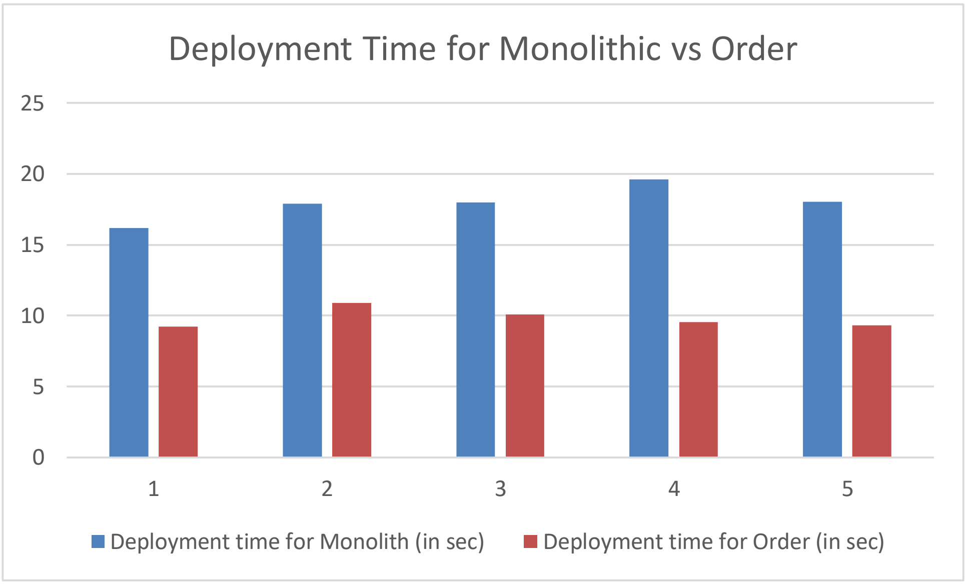

Figure 5:Deployment Time for Monolithic vs Order

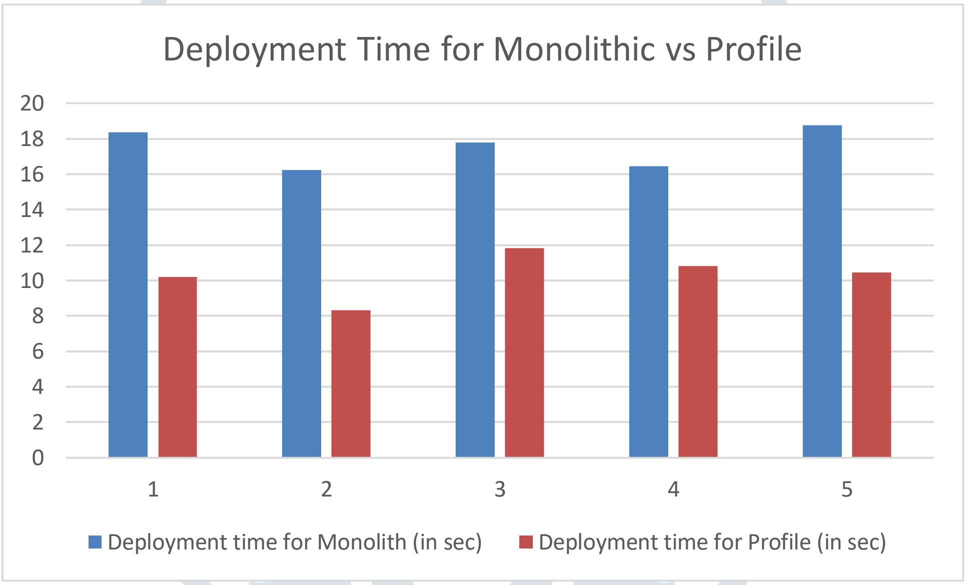

Figure 6:Deployment Time for Monolithic vs Profile

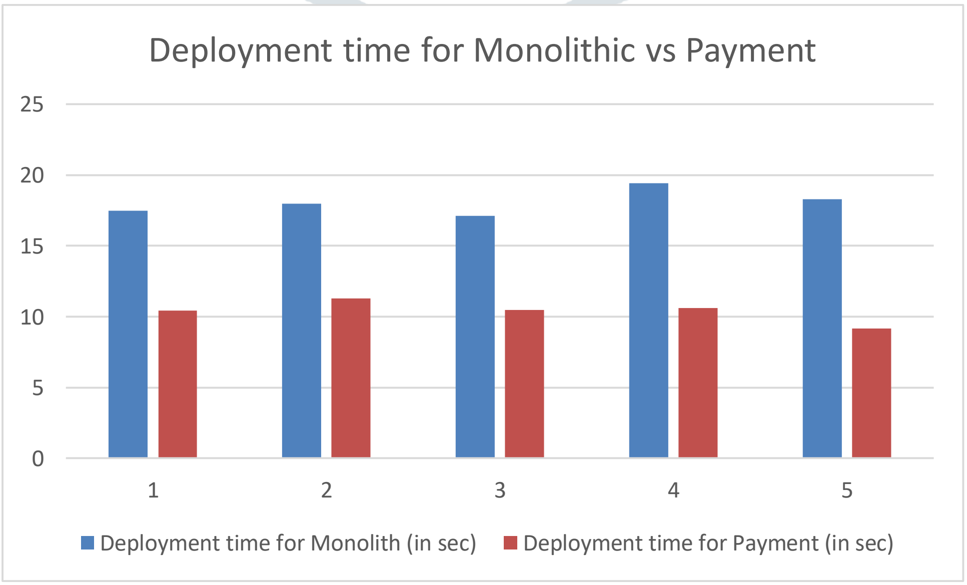

Figure 7:Deployment time for Monolithic vs Payment

From above table, we can conclude that the even if on different instances, the time required for deploying monolithic application always exceeds than that of application build on Microservice Architecture, resulting in less downtime in case of Microservice implemented application as compare to Monolithic application.

VIII. CONCLUSION

We have created a system which will help us for comparing downtime and complexity of both monolithic and Microservice application which leads to understand the benefits and advantages of microservices over monolithic architecture.

REFERENCES

[1]. S. Newman, Building Microservices. O'Reilly, 2015.
[2]. https://dzone.com/articles/benefits-amp-examples-of-microservices-architectur
[3]. Wilhelm Hasselbring and Guido Steinacker, "Microservice Architectures for Scalability, Agility and Reliability in E-Commerce"
[4]. "Information system integration," W. Hasselbring, Communications of the ACM, vol.43, no. 6, pp. 32–36, 2000
[5]. IEEE Multimedia, vol. 9, no. 1, pp. 16–25, 2002. "Web Data Integration for E-Commerce Applications,".
[6]. M. Abbott and M. Fisher, The Art of Scalability, 2nd ed. Addison- Wesley, 2015
[7]. J. Waller, N. C. Ehmke, and W. Hasselbring, "Including performance benchmarks into continuous integration to enable DevOps," ACM SIGSOFT Softw. Eng. Notes, vol. 40, no. 2, pp. 1–4, Mar. 2015
[8]. 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" 10th Computing Colombian Conference (10CCC), pages 583–590, Sept 2015.
[9]. Dharmendra Shadija, Richard Hill and Mo Rezai, and. 2017. "Towards an Understanding of Microservices". 23rd IEEE International Conference on Automation and Computing. IEEE Computer Society, Huddersfield, UK.
[10]. Microservices: Granularity vs. Performance by Richard Hill , Dharmendra Shadija and Mo Rezai
[11]. Thones. 2015. Microservices. IEEE Software 32, 1 (2015)