MICROSERVICES: DIVIDE AND CONQUER

Ruchita Garde
George Mason University, rgarde@gmu.edu

Microservices is an architectural concept of building IT systems using a divide-and conquer approach right from the design to deployment stages of system development life-cycle. This paper is conceptually based on the article 'The Hidden Dividends of Microservices' by Tom Killalea [1]. The benefits of using microservices are varied and deep including resilience, agility, scalability and developer productivity. The better you are at embracing microservices, the more surprised you will be as you discover the hidden benefits. Established organizations trying to look out for options better than monolithic architectures, as well as startups wanting to expand-microservices have something for everyone.

INTRODUCTION

Microservices architecture is a way of building huge systems by dividing the complexity of the software into small concern-specific modular services which are developed and deployed independently. It is a practice of employing service-oriented architectures at a small level of granularity [2]. Each service module is tightly bound around its share of responsibility, while the modules are dissociated from each other to support a high level of modifiability. Microservices are a modern interpretation of Service- Oriented Architectures (SOA) that has happened after the introduction of DevOps and is becoming the standard for building continuously deployed systems [3].

WHAT'S WRONG WITH MY MONOLITHIC SYSTEM?

Classic SOA architectures mean huge, bulky and intimidating codes since a monolithic system is made up of a single platform with programs for both user interface design, as well as persistence techniques. Monolithic systems are less granular, which means the absence of all the advantages that modularity has to offer. Projects with long lifecycles are in need of constant upgradation. In such a case, if making changes in one component of a huge network of inter-related of components escalates to the need of changing many other components, it may spell failure for the entire system. Even if we consider a sunny day situation, the modification is going to take a lot of time needing constant inter-team communication leading to slow deliveries. Other factors hindering the agility is low developer productivity. New team members take time to become productive because of the time they require to understand the entire system at once before actively contributing towards actually developing the code.

HOW CAN SHIFTING TO MICROSERVICES HELP ME?

The basic idea behind microservices is decomposing a system into smaller deliverables, with each deliverable providing a close knit service but completely decoupled from other services. Each service has the freedom from control of other services. Services are also unaware of the data-storage mechanisms, languages and such details of other services. Each microservice manages its own database. This not only provides development teams with a more decentralized approach to building software, it also allows each service to be deployed, re-built, re-deployed and managed independently. [4] Such an architectural approach increases the speed of business development. Also, the strong cohesive nature of services ensures that the issues which may arise get resolved quickly. This is because changes made to one service will not mandate that any change be made to another service, which ensures that the failure of one module does not spread its ill-fate to any other service. Also, the specific engineering team working on a service will have complete ownership of the service. The benefits of this liberty provided to the team include no requirement of time consuming co-ordination between different teams. The team has the power to disapprove of a service if it is not delivering the way it was intended to. The team can also devise test conditions beforehand, ensuring the tests prove more fruitful since the rate of correction of failure causing issues is more than what would have been in a monolithic architecture.

I AM JUST A STARTUP

If you are an organization which is nascent with few or no legacy architectures that the business or engineering resources are accustomed to, then microservices would be a good way to start. An important reward this approach provides to newbies is the ability to learn and quickly recover from mistakes, try out revolutionary ideas and get immediate feedback. Firstly, the owner team of the service can make data storage mechanism choices and User interface design choices without feeling the need to follow any protocols common to peer services, since there aren't any. The team closely monitors its service, and is at a very good position to get user feedback. Based on this response, changes and improvements can be made to persistence systems and UI without other services being affected. The team can develop its own debugging, fault-solving techniques. Multiple implementations of the same service module can be designed and tested with customers, and the most optimum variant can be eventually used, the decision resting with the service team. In cases where failure or performance prediction is not possible for budding developers, microservices architecture comes to great help. On this scalability benefit, Atul Saini in his blog says, 'Since an application is composed of multiple micro services which share no external dependencies, scaling a particular micro service instance in the flow is greatly simplified: if a particular microservice in a flow becomes a bottleneck due to slow execution, that Microservice can be run on more powerful hardware for increased performance if required, or one can run multiple instances of the Microservice on different machines to process data elements in parallel.' [5]

AM I TRULY INTO MICROSERVICES?

Like many other times, when the organizations feel that they invariably need to adopt the current hot market trend if they have to survive in the competition, hasty adoption of a new conceptual practice without completely understanding its nuances will fail to provide the dividends it was supposed to. Migrating to a new IT culture from the historic values your organization has been following takes time.
It is important for organizations to keep a constant check on themselves to make sure they have embraced microservices the way they were supposed to, and not stuck somewhere in between. You are not doing it right if you require co-ordination between teams or between services for deployment.
Make sure neither your services nor their servicing teams have dependability on each other. If the organization is imposing protocols across the services horizontal, then you are not going to get the results you expected.

THE OTHER SIDE

It can be really difficult for the people of an organization following a monolithic architectural system to make the transition to microservices. As Jeff Genender correctly said, 'One challenge in implementing microservices has been getting organizations to understand the concept that an application becomes completed, distributed and not monolithic. Many developers still believe applications are deployed in a single directory or as a fully packaged container. It's sometimes tough to get people to break that a paradigm.' [6]

CONCLUSION

The benefits of using microservices are varied and deep including resilience, agility, scalability and developer productivity. Microservices architecture is a way of building huge systems by dividing the complexity of the software into small concern-specific modular services which are developed and deployed independently. For the organizations implementing microservices, there are a lot of rewards which monolithic SOA otherwise cannot provide. The migration from legacy architectures can be difficult for many, but it surely is worth the effort.

REFERENCES

[1] Killalea Tom, "The Hidden Dividends of Microservices", *acm queue*, Vol. 59, No. 8, August 2016.
[2] Thomas Anne, http://searchsoa.techtarget.com/feature/Microservices-is-more-than-a-buzzword
[3] https://en.wikipedia.org/wiki/Microservices
[4]http://searchsoa.techtarget.com/definition/microservices
[5] Atul Saini, https://www.fiorano.com/blog/microservices/microservices-architecture-scalability-devops-agile-development/
[6] Jeff Genender , http://searchcloudapplications.techtarget.com/feature/How-microservices-bring-agility-to-SOA