A Review on Shifting to Microservices

Vaibhav Nigam, Vallidevi Krishnamurthy, K K Nagarajan

Abstract— Microservices are seen as an architecture style for scalable, fast evolving cloud applications. In this paper, the ways in which a monolithic architecture can be converted to a micro-service based architecture and scaled to a larger size is highlighted. The architecture and the performance monitoring procedure of a microservice is also discussed in this paper.This could be used by engineers working on relatively large scale projects to switch their monolithic projects to microservice based software which could be beneficial in many ways.

Keywords: Cloud, micro-services, SOA, webservices, bigdata

1. INTRODUCTION

A monolithic application describes a single-tiered software application in which the user interface and data access code appears as a single program from a single platform. A monolithic application is self-contained, and independent from other computing applications. An SOA is an architectural pattern in computer software design in which application components could be fetched as services via a communications protocol, typically over a network. Or those components could even be offered as services to some other software development. Micro-services are a variant of SOA architectural style that structures an application as a set of loosely coupled services, where the services are fine grained and the protocols are light weighted.

1.2 Why switch to Micro-service based structure?

There are many reasons why the developers of an application/software must consider breaking down their monoliths to smaller micro-services. The reasons include, Evolutionary Design, Decentralized Data Management, Componentization, More focused tasks in hand for developers, Well structured, Better fault isolation, Easier Scaling, Complement cloud activities [1].

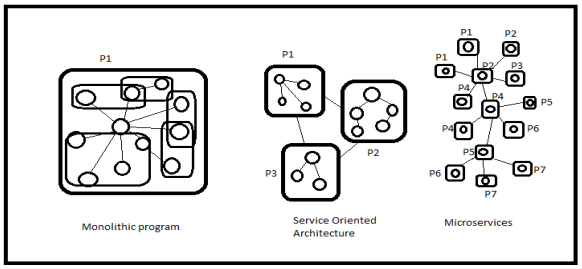

Figure 1: Structural Difference between Monolithic, Web services and Micro-Service

Revised Version Manuscript Received on April 05, 2019.

Vaibhav Nigam, SSN College of Engineering, Kalavakkam, Tamil Nadu, India.(E-Mail: vaibhav16115@cse.ssn.edu.in)

Vallidevi Krishnamurthy, SSN College of Engineering, Kalavakkam, Tamil Nadu, India.(E-Mail: vallidevik@ssn.edu.in)

K K Nagarajan, SSN College of Engineering, Kalavakkam, Tamil Nadu, India.(E-Mail: nagarajankk@ssn.edu.in)

Monolithic programs work only on one platform (in Figure 1, its P1), while web-services are platform independent (in Figure 1, its shown as P1, P2, P3), where functionality in each platform can be subdivided to multiple services. In micro-services, there can be any number of platforms (in this case 7 platforms), where none of the services can be divided.

2. RESEARCH QUESTIONS

- Q1. Why should a successful IT company switch over to micro-service when there is no issue currently?
- Q2. At what point should a monolithic program be converted to a micro-service based structure?
- Q3. How to handle the data when switching to micro-service based architectural design?

To address the above question, few case-studies are discussed below.

2.1 Why a switch-over is needed?

Walmart successfully changed its failing architecture with microservices. [1]

As the literature revealed, Walmart initially had problems handling 6 million page-views per minute. They had failed on Black Fridays for 2 years in a row. Migrating to microservices yielded the following results for Walmart such as, Conversions up by 20% overnight, Mobile orders up by 98% instantly, No downtime, Saved 40% of the computing power and experienced 20-50% cost savings overall.

Spotify builds outstanding user-experience with micro-services. [1]

Initially when Spotify had a monolith project structure, their business bloomed. But as the number of users increased drastically, their application could not scale up. So they now have adapted to the micro-service architecture and are now able to scale up their software for 100s of millions of users.

Amazon adopts to micro-services [1]

In 2001, the amazon.com retail website was a large architectural monolith. It was built with multiple tiers, and those tiers had many components in them. But they were coupled together very tightly, and behaved like one big monolith. Due to this coupling, a team had to convince many other teams for updation of libraries, if they were to incorporate a new feature and that lead a lot of wastage in time. Amazon first moved to Service Oriented Architecture based application, to deal with the tight coupling issue. They experienced its benefits and have now slowly moved to a micro-service based architecture. This now enables them to handle 50 million deployments a year with ease. From these

case studies, it's safe to say that shifting to a micro-service based architecture can be very beneficial if done the right way and that answers Q1.

2.2 When to switch over to micro-service based architecture?

Any software architecture cannot be converted to a micro-service based architecture. In certain scenarios, the micro-service based architecture on a project may lead to an increase in response time as output from an arbitrary service may have to act as the input to another service and there might be latency caused in that transfer of data.

For example, in software applications that are used by banks, where there is a need for real time response to actions, micro-service based architecture may not be the best option. Hence, the conversion from a monolithic program to a micro-service or a web service based applications must be done only in the following conditions.

- The number of micro-services (individual, loosely coupled functionality) is greater than an arbitrary number depending on the kind of software itself.
- The amount of data needed to be stored by the software/project is greater than what can be stored in at least a single mainframe system.
- Each micro-service can have its own database structure that is cohesive with that of the other micro-services.
- Slight latencies can be dealt with.

This answers Q2.

2.3 How to handle the data when switching over to micro-services?

Once the amount of data that is involved grows over a certain limit, it simply cannot be handled with simple RDBMS systems that were used in monolithic programs/ software applications, as that much of data cannot be processed efficiently within the required time frame. To deal with this issue, the way data is handled has to be changed. Big product based companies that fall into such categories have changed their file systems in order to deal with this issue.

Hadoop Distributed File System [2]

The Hadoop Distributed File System (HDFS) is a distributed file system designed to run on commodity hardware. It has many similarities with existing distributed file systems. However, its differences from the other distributed file systems are significant. HDFS is highly fault-tolerant and is designed to be deployed on low-cost hardware. HDFS provides high throughput access to application data and is suitable for applications that have large data sets. Companies that make use of this file structure include Facebook, Twitter, Alibaba, Microsoft and many more.

How does Cloud work into all of this? [3]

The cloud is ideally suited to provide the big data computation power required for the processing of these large parallel data sets. Cloud has the ability to provide the flexible and agile computing platform required for big data, as well as the ability to call on massive amounts of computing power (to be able to scale as needed), and would be an ideal platform for the on-demand analysis of structured and unstructured workloads.

Practical implementation steps for Distributed Data Handling [8]

The developer team will have to first decide on what functionality can be separated out from the existing monolithic structure. Once that is decided, the team will have to fix locations where they can have their servers placed for distributing the data system. Then the file system can be changed to HDFS which will render the path for the scalable data set. Thus, the data could be distributed, with a fault-tolerant, robust micro-service based architecture to enable the software to scale up and work efficiently. This method can help solve the aforementioned problem of scaling and the Q3 has been answered.

3. MORE ABOUT MICROSERVICES & RESULTS

Microservices can be expanded in X, Y and Z axis [10]. If the number of functionalities are increased, then it is said to be as expanding in X axis. If the number of users of each service is scaled, then it is expanding in Y axis. If the application is scaled at business level, then it is called as z-axis scaling.

The different categories of services in microservices are Platform as a service (Paas), Software as a service (Saas), Infrastructure as a service (Iaas), Data as a service (Daas), Backend as a service (Baas) or Mobile backend as a service (MBaas) [10].

Microservices are not composed through orchestration, and are composed only through choreography. Different compositional patterns that are available in microservices are Aggregator, Proxy, Chained, Branch Microservice and Shared Resource pattern. Shared resource pattern is the most used pattern among the organizations. Chained pattern should have minimal number of services in its chain to overcome efferent coupling.

4. RESULTS – COMPARING MICROSERVICES WITH WEBSERVICES

Communication among the web services happens via an enterprise service bus. So it works in the concept of share-as-much-as-possible. Rather if the number of communication links increases in the case of microservices, then there will arise a high efferent coupling. So the choreography between the functional services is to be minimal. The interactions should be only among the functional and infrastructure services. In the case of microservice, if the service granularity is too fine, then the number of interactions to fulfil a single business task will be more and this leads to adding up of response time along with the execution time of the services. Hence, as stated by Sam Newman [9] while developing a microservice, it is advised to start with a small number of larger services first.

The opensource microservice application was monitored using the Kieker framework which was discussed in [11]. Microservices are costlier as they have to maintain different servers for different business applications. However, deployment of a microservice is easier as it is light weightwhen compared to webservices.

5. CONCLUSIONS

With these discussions, it could be concluded that, shifting from a monolithic program structure to a microservice based architecture would be a great choice if the application/software if the scope of the software is diverse enough for it to be split in the first place. It could be assured that, scalability and fault-tolerance are the major benefits of the systems that adapt to micro-service based architecture development.

REFERENCES

1. Vural H., Koyuncu M., Guney S. (2017) A Systematic Literature Review on Microservices. In: Gervasi O. et al. (eds) Computational Science and Its Applications – ICCSA 2017. ICCSA 2017. Lecture Notes in Computer Science, vol 10409. Springer, Cham
2. Dmitry Savchenko, Gleb Radchenko, "Microservices validation: Methodology and implementation", CEUR Workshop Proceedings. Vol. 1513: Proceedings of the 1st Ural Workshop on Parallel Distributed and Cloud Computing for Young Scientists (Ural-PDC 2015), 2015.
3. https://dzone.com/articles/benefits-amp-examples-of-microservices-architecture
4. https://hadoop.apache.org/docs/r1.2.1/hdfs_design.html
5. https://www.ibm.com/blogs/cloud-computing/2014/05/07/hadoop-relate-cloud/
6. https://mapr.com/blog/using-microservices-evolve-beyond-data-lake/
7. https://blog.equinix.com/blog/2017/07/11/7-things-you-need-to-know-about-microservices/
8. https://aptude.com/blog/entry/benefits-and-challenges-of-using-microservices-with-big-data- applications/
9. Sam Newman, Building Microservices - Designing Fine-Grained Systems, O'Reilly Media, 2015.
10. Microservice Architecture from Tutorials point- Simply easy learning.
11. Saman Barakat, Monitoring and Analysis of Microservices Performance