University of Essex | Online

Literature Review

Application of Microservice Architecture in the Development of Web Applications

MSc Cyber Security

University of Essex Online

Marzio Hruschka

Student ID: 12684419

Research Methods and Professional Practice

Unit 7

Tutor: Karen Outram

Date of Submission: 25/04/2022

Table of Contents

1. Introduction

Microservices is a relatively new architectural approach influenced by service-oriented computing and is considered the latest trend in software service design, development, and delivery. Within a microservice architecture, an application is isolated into a collection of loosely coupled autonomous services that work together to perform business functions. They can be written in different languages and communicate with each other through APIs (Dragoni et al, 2017).

This new architectural design has gained a lot of popularity in recent years, and many new applications are being developed using a microservice approach. According to a survey conducted by Vailshery (2021), 34% of over 4200 participating organisations worldwide are already utilising a microservice approach within their application and 37% have adopted the approach partially. Additionally, a survey conducted by IBM Market Development & Insights (2021) has found that more than 84% of surveyed current users and 74% of non-users of microservices believe that microservices offer many benefits to development teams and that a microservice architecture creates better collaboration among team members.

Since microservices is a new approach to application architecture, a literature review of current scholarly sources will be examined to discuss the application of microservice architecture in the development of web applications and provide a holistic academic perspective on the subject. A comparison to the more traditional monolithic approach will be established to provide an outline of current knowledge of the challenges and advantages of both designs while highlighting key differences. The banking industry will be used as an example to discuss a practical implementation of the architectural approach.

2. Comparison of Architectural Approaches

2.1 Monolithic Architecture

Common development languages such as Python and Java enable applications to be broken down into individual functions and classes. However, these languages are traditionally designed to create a single enclosed system, also called a monolith, whereby the individual code blocks and modules are all utilised within a single, self-contained unit (Dragoni et al, 2017).

Dragoni et al (2017) establish a comparison of both the traditional monolithic and the microservice approach, highlighting the limitations of the former and presenting an outlook of how the innovations of microservices may overcome these challenges. The authors define the monolith as “a software application whose modules cannot be executed independently”. As Dragoni et al describe, within a monolithic architecture, the code base is often rather large and interconnected. This makes it difficult to troubleshoot issues and maintain the system in the long run due to its complexity. Furthermore, third-party libraries and services, as well as the underlying programming language cannot be easily changed or updated without making major changes to the overall system. Once the application’s technology has been decided on, the developers are commonly bound to use the same tech stack throughout the application’s lifecycle.

Another highlighted limitation is the deployment rigidity of the system. Whenever a new feature or update is implemented, the whole system needs to be taken offline to be redeployed. This also makes the testing process strenuous, as the complete system has to be tested and validated due to reliance on a single code base. Lastly, a monolithic application offers limited scalability and a non-efficient use of resources.

The growth of users can only be handled by creating new instances of the whole application and splitting the load amongst the instances. However, often an increase in traffic only strains certain modules of the system which cannot be individually rescaled in an enclosed monolithic system (Dragoni et al, 2017; Al-Debagy & Martinek, 2018).

Ponce et al (2019) conducted a study to identify and analyse various techniques to migrate a monolithic architecture to that of a microservice. Within the study, the authors highlight typical issues with monoliths that can be either technical (e.g. highly coupled system and maintenance issues) or business-related (e.g. extended time-to-market, lower productivity and efficiency). Ponce et al agree with the presented challenges, however, they state that it may be a good idea to start developing an application using a monolithic approach, due to it allowing the developers to explore the limitations and complexity of a system and its components. Once these limitations have been reached, Ponce et al (2019) suggest that the identified issues may become apparent and that migration towards a microservice approach represents the best option to overcome these challenges.

2.2 Microservice Architecture

Zimmermann (2017) provides a cohesive overview and review of the microservice approach. The author argues that the microservice architectural style can be considered a natural extension of SOA (Service-Oriented Architecture), which emphasises the independence and lightweight nature of services. Each microservice is centred around a business capability, runs in its own process, and connects with

other microservices in the application via lightweight protocols (e.g. using RESTful APIs).

Dragoni et al (2017) present the microservice approach as an innovation that helps cope with the mentioned issues of monoliths. The authors define microservice architecture as “a distributed application where all its modules are microservices”. By structuring individual microservices around a single business functionality, the code base of each service remains small and lightweight. Deployments and testing can be done on the independent microservice, and fixing bugs only involves the modules where the bugs occur. Therefore, whenever a new feature or update is released, only the affected service needs to be rebooted, whilst the entire application may remain operational. Moreover, microservices can be containerised and thus offer a high degree of flexibility for integrations and programming languages. Regarding scalability, services can be rescaled independently depending on their respective load.

As described by Ponce et al (2019) many large systems have migrated from a self-contained and interdependent monolith to a more lightweight and loosely connected microservice application in recent years. This is explained by the market’s rapid demand for new features which necessitates changes that cannot be met in a traditional monolith.

2.3 Pains and Challenges of Microservices

While much of academic literature describes microservices' benefits over the use of a monolithic approach, not many sources go in-depth regarding the challenges and potential pitfalls that an adaptation or migration towards a microservice architecture could bear. The microservice approach does solve many of the aforementioned issues of the traditional monolith, however, it is important to be mindful of these disadvantages as well.

Dragoni et al (2017) do highlight two important downsides. Firstly, due to microservices running as independent modules and relying on messaging protocols such as HTTP APIs to interact with each other, the communication over the network could hurt performance, reliability, and availability. The latency of a network is much greater than that of in-memory calls that are used within a monolithic application. Quality and performance of communication deteriorate when having to go through a network and having even a single service request fail could mean that the whole system may not be accessible.

Secondly, more overhead might be required regarding security concerns. As most data is being transmitted between services, it is necessary to ensure that all data-in-transit is sufficiently encrypted. Furthermore, due to the nature of flexible third-party integrations, additional security mechanisms need to be implemented to ensure that consumed data is stored safely and that its integrity is guaranteed. Overall, the microservice architecture provides a greater attack surface when compared to the monolithic paradigm. Application internals are accessible via the network and are thus now accessible to external actors, including malicious ones (Dragoni et al, 2017).

Jamshidi et al (2018) address the challenges of microservices in more detail. Microservices are a relatively new concept and as such expertise in this paradigm might still be limited. Due to the increase in popularity, microservices are increasingly likely to be deployed in circumstances where the costs substantially outweigh the advantages. Microservices are not the right approach in all use cases. Some applications make complete sense to be developed in a monolithic approach. This does not mean that the application should not be constructed in a modular fashion, however, the individual modules do not always have to be completely isolated.

Furthermore, as the size of a microservice application grows, the overhead for efficiently monitoring and managing its resources such as containers and virtual machines increases as well. Microservices may need to be deployed across multiple availability zones, resulting in a multitude of logs, which makes deciding on time-sensitive matters very difficult. Having autonomous teams work on separately deployed services may present challenges as well. Whilst these teams make independent decisions on their own service, they might not consider the holistic goals of the application and organisation (Jamshidi et al, 2018).

2.4 Performance Comparison

In general, research seems to agree on the theoretical benefits of implementing a microservice architecture over that of a monolithic one. However, when looking at studies that have examined the performance of both approaches side-by-side, the results are contradictory:

In a study by Singh & Peddoju (2017), the performance of a monolithic application was compared to that of a developed microservice solution using indicators such as

response time, time to deployment, and throughput. The paper concluded that the monolithic application could be outperformed by the microservices-based application by reducing the response duration, offering a higher degree of scalability, as well as increasing the throughput.

However, a study by Al-Debagy & Martinek (2018) came to a different result. The authors carried out a similar comparison by evaluating specific configurations of microservices' applications and measuring performance deviations in comparison to monolithic architecture. The outcome exhibited stark differences from the study conducted by Singh & Peddoju by concluding that monolithic architecture scored higher performance in throughput with no significant difference in load testing.

Building on the previous studies, a third study conducted by Ueda et al (2016) compared both architectures but concentrated more on resource consumption. The study observed a significant resource overhead for the microservice approach, e.g., consuming Node.js libraries 4.22 times longer than the monolithic version. Microservices performance was measured to be up to 79.2% lower than the monolithic equivalent, thus supporting the claims that monolithic architecture outperforms microservices architecture.

3. Practical Application within the Banking Industry

The banking industry is experiencing pressure with the market's high pace of demand for new features and changing requirements. Legacy architecture within banking comprises monolithic structures and mainframes that prevent an easy adaptation of innovation. To stay competitive, banks increasingly explore the microservice paradigm as a solution to ensure flexibility and scalability. However, legacy mainframes are

complex systems carrying crucial information and often need to meet compliance of a central governance body before being able to be migrated (Hayretci et al, 2021).

Bucchiarone et al (2018) present an interesting case study of how Danske Bank migrated its Forex Core Application from monolithic to microservices in a gradual way that did not compromise the bank's daily operations. Business functionalities were evaluated on a case-by-case basis, and only if deemed isolated and big enough or utilised by other business capacities, would result in migration to a new service. Due to the required clearances and data criticality, Danske Bank opted to keep its legacy mainframe in place and instead integrated the microservices into it. However, over time, the functionalities of the mainframe will be replaced using isolated microservices.

To overcome the issues concerning monitoring and management overhead, the team opted to utilise the containerisation solution, Docker and its orchestration tool, Docker Swarm. The team at Danske Bank found that with the introduction of microservices, scalability, clear segregation of functionality and high coherence within the application were successfully achieved. However, new challenges such as concurrency handling, fault-tolerance, and monitoring became apparent which did not exist in the legacy monolith solution (Bucchiarone et al, 2018).

Danske Bank's approach to implementing containers and an orchestration tool coincides with the technologies recommended by Jamshidi et al (2018). Tools such as Kubernetes and Docker Swarm automate container allocation and management tasks and provide an abstraction for the underlying physical or virtual infrastructure. These solutions may help overcome the highlighted challenges of increased overhead and impact on efficiency.

4. Conclusion

In a market that demands rapid changing user requests and features, microservices provide a solution to remain agile and accommodate growth. With microservices, organisations can meet the market's demand for innovation and adaptation while keeping business functionalities segregated, as well as ensuring deployment and testing processes are streamlined. However, challenges such as security concerns, performance and limited expertise need to be kept in mind. It should also be emphasised that the microservice approach does not necessarily constitute the best solution for every application.

Companies already seem to be adopting microservice architecture at a quick pace (IBM Market Development & Insights, 2021; Valishery, 2021), however, there is an apparent gap within academic research concerning the explicit pains of the introduction or migration of a microservice application. Whilst there are numerous sources on the theoretical benefits of implementing the paradigm, the practical challenges are not highlighted as extensively. This lack of literature could be explained by researchers not having easy access to live large-scale microservice implementations and the challenges that might arise with them.

Future areas of studies could for instance include creating a framework for the practical migration of live production systems without impacting day-to-day operations, as well as crucial tools to help ensure a successful implementation.

List of References

Al-Debagy, O. & Martinek, P. (2018) ‘A Comparative Review of Microservices and Monolithic Architectures’, *2018 IEEE 18th International Symposium on Computational Intelligence and Informatics (CINTI).* Budapest, Hungary, 21-22 November. Piscataway: IEEE. 149-154.

Bucchiarone, A., Dragoni, N., Dustdar, S., Larsen, S.T. & Mazzara, M. (2018) From Monolithic to Microservices: An Experience Report from the Banking Domain. *IEEE Software 35*(3): 50-55. DOI: https://doi.org/10.1109/MS.2018.2141026

Dragoni, N., Giallorenzo, S., Lafuente, A.L., Mazzara, M., Montesi, F., Mustafin, R. & Safina, L. (2017) ‘Microservices: Yesterday, Today, and Tomorrow’, in: Mazzara, M. & Meyer, B. (eds) *Present and Ulterior Software Engineering*. Cham: Springer International Publishing. 195-216.

Hayretci, H.E. & Aydemir, F.B. (2021) ‘A Multi Case Study on Legacy System Migration in the Banking Industry’, in: La Rosa, M., Sadiq, S. & Teniente, E. (eds) *Advanced Information Systems Engineering*. Cham: Springer International Publishing. 536-550.

IBM Market Development & Insights (2021) Microservices in the enterprise, 2021: Real benefits, worth the challenges. Available from: https://www.ibm.com/downloads/cas/OQG4AJAM [Accessed 12 April 2021].

Jamshidi, P., Pahl, C., Mendonça, N.C., Lewis, J. & Tilkov, S. (2018) Microservices: The Journey So Far and Challenges Ahead. *IEEE Software 35*(3): 24-35. DOI: https://doi.org/10.1109/MS.2018.2141039

Ponce, F., Márquez, G. & Astudillo, H. (2019) ‘Migrating from monolithic architecture to microservices: A Rapid Review’, *2019 38th International Conference of the Chilean Computer Science Society (SCCC).* Concepcion, Chile, 4-9 November. Piscataway: IEEE. 1-7.

Singh, V. & Peddoju, S.K. (2017) ‘Container-based Microservice Architecture for Cloud Applications’, *2017 International Conference on Computing, Communication and Automation (ICCCA).* Greater Noida, India, 5-6 May. Piscataway: IEEE. 847-852.

Ueda, T., Nakaike, T. & Ohara, M. (2016) ‘Workload Characterization for Microservices’, *2016 IEEE international symposium on workload characterization (IISWC)*. Providence, USA, 25-27 September. Piscataway: IEEE. 1-10.

Vailshery, L. (2021) Organizations' adoption level of microservices worldwide in 2021. *Statista*. Available from: https://www.statista.com/statistics/1233937/microservices-adoption-level-organization/ [Accessed 12 April 2021].

Zimmermann, O. (2017) Microservices Tenets. *Computer Science-Research and Development 32*(3): 301-310. DOI: https://doi.org/10.1007/s00450-016-0337-0