Microservices: A Review of the Costs and the Benefits

Ahmed Elfatatry
Information Technology Department
Alexandria University
Alexandria, Egypt
elfatatry@alexu.edu.eg

Abstract—**This work is concerned with analyzing the merits and the costs of Microservices. Following the hype associated with a new technology may result in more problems rather than solutions. The Microservices approach offers many benefits in terms of flexibility and scalability. However, rushing to use Microservices without balancing the situation may add unnecessary complexity and there is a possibility that the costs may outweigh the benefits. The key question is: are Microservices in their current form solving more problems than they create? In this paper, we analyze the benefits and the costs of switching to Microservices. The aim is to support the decision of whether to move to Microservices or not based on the evaluation of the advantages and the disadvantages. The main contribution of this work is the provision of a clear picture of the costs and benefits of the technology to help decide if and when a switch to Microservices is the correct choice.**

Keywords- Microservices; Flexibility, Scalability, Software Engineering.

I. INTRODUCTION

Software Engineering has a long history of proposing ways to deal with change. Flexibility is a desirable software attribute, especially in business systems [1]. The ability to change a system as a result of changing requirements with minimum cost has been at the heart of Software Engineering solutions [2] . Targeting low coupling, high cohesion, modularization, and separation of concerns are just a few examples.

The "service thinking" has been a shift in how software is created and delivered. The core of such thinking is that the focus should be on how to consume a functionality rather than the means by which the functionality is produced [3]. In software terms, it is the decoupling of the producer from the product. The Service Oriented Architecture (SOA) has been an early implementation of such concept.

Micorservices are software components where independence of development and deployment is a key concern [4]. The concept of loose coupling is fundamental to the idea of Microservices. Better flexibility can be achieved if a system is built using independent services.

In this work, we provide an analysis of the costs and the benefits of Microservices with respect to achieving flexibility. The aim is to support the decision of switching to Microservices.

This paper is structured as follows. Section 2 examines the concept of Microservices and highlights the benefits of applying such thinking. In Section 3, the Microservices model is compared with the monolithic model in the context of flexibility. Section 4 analyzes the inherent problems of Microservices. The challenges of and future of Microservices are discussed in Section 5. Finally, the conclusions are presented in Section 6.

II. PROBLEM STATEMENT

Microservices architecture is a relatively new architecture which originated in the industry. While there is a great interest in the academia, however, there is an obvious gap between the academia and the industry concerning the topic. Few experience reports by the industry are available, and less practical solutions from the academia [5]. Research efforts usually focus on a single aspect of Micorservices such as migration from legacy systems, architecture, security, database heterogeneity, or service patterns. Other research works highlight only benefits [6], or otherwise only disadvantages. There is a gap in the literature concerning studies that balance the costs versus the benefits.

III. WHAT ARE MICROSERVICES?

The Microservices approach is another way to think about how to build software applications. The approach advocates building the applications as suites of small, independently deployable services, each running its own process. There is no universally accepted definition of Microservices. The independence of Microservices is a key design issue. A service has to be independently deployable. The reason is that this issue has an impact on managing large scale deployment. Each service would be independently developed as a self-contained product with its own complete team. Microservices are built to serve a specific context. Services are built around business capabilities [7]. Related functionalities are combined into a single business capability, and each Microservice implements one such capability [7]. The development team would include a user interface person, a database person, and a business logic person. A team is usually responsible for the whole life cycle of a Microservice [8].

Having described what Microservices are, in the following we discuss the benefits of such approach.

- It is possible to release functionality faster. The reason is that it is not needed to wait until it is possible to release the whole system. Bringing changes into production rapidly is a priority for any business. The more an application is broken down into smaller components, the easier it is to deal with changes. Currently, this is not the case with most monolithic applications [9] [10].

 ISBN: 978-1-61208-701-6

- Braking a system into smaller components supports flexibility. The consumer of a specific functionality may choose from a number of providers that provide the same functionality with different non-functional attributes. The more the software is loosely coupled, the easier it is to engage with open source, provided that there is a license. If a company needs some functionality to be incorporated into the systems from open source components, it has to decompose the system into smaller loosely coupled components.
- Independent scaling. Only the parts of the application that need to be scaled up can be assigned the required resources. There is no need to upgrade the whole infrastructure only to serve selected parts of the system [11]. The result is efficient use of resources. Parts of the system that need more computing power can be assigned the needed resources without having to scale up the whole system [12].
- It is easier to focus on security wherever it is needed. More sensitive services could be put into more protective zones. Less sensitive services that require less protection can be assigned the appropriate resources.
- Each service can be built using the best and most appropriate tool for the task. It can be possible to move parts of the system to the cloud [13]. A company may decide to put some components on the cloud to be managed by specialized competencies. Whether or not the decision is to use multiple technologies in a system, there is a possibility to do it if needed [13].
- Redundancy. Usually, it is assumed that redundancy should be avoided. In a Microservice design, redundancy is a classic way of increasing resilience. Microservices can help implementing this concept more easily.

IV. MICROSERVICES VERSUS MONOLITHS

In a monolithic appication, modules cannot be executed independently. Any change in one module of a monolith requires rebooting the whole application. Scalability is usually a problem in monolithic applications. Often, the entire application does not need to be scaled up. Only a subset of the modules need to be scaled up. The usual strategy for handling such situation is to create new instances of the entire application.

Typically, when monolithic architectures are exposed to a growing load, it is difficult to locate which components of the system are actually affected, since the system runs within a single process. This means that although only a single component may be experiencing load, the whole monolith will need to be scaled up. This will be the only solution even if it is known which component is experiencing the load, as it is difficult to scale it in isolation

V. MICROSERVICES VERSUS SOA

Microservices are mainly focused on application architecture, but they may have some elements that can be taken to the enterprise level. This depends on the size of the enterprise. SOA is an enterprise level concept. SOA is on the enterprise scale while Microservices is on the application scale. In short, a Microservice is a component while SOA is an architecture.

In the traditional SOA, organizations would buy and deploy an Enterprise Service Bus (ESB) and then deploy their individual services on that ESB. But if more scalability is needed, then, the entire ESB has to be scaled up [13]. The Microservices advantage here is that individual services can be scaled up. As Martion Fowler points out, the difference is the shift from the intelligence that is built into the transport layer to having the end points more intelligent and the pipes being a little less intelligent [7].

VI. PROBLEMS OF MICROSERVICES

As stated in [11], the first disadvantage of Microservices is its name. The goal of Microservices is to decompose an application in order to facilitate development and deployment of agile applications. Building small services is not the goal of Microservices, but rather facilitating agile development.

Although individual services may be very simple, there is an increase in complexity as a result of having communications between different components. Distributed systems are more complex compared to monolithic systems. In addition, managing running services is more complex compared to monolithic services.

Partitioning a database across a number of different Microservices makes it difficult to implement some business transactions that can be implemented much easier in monolithic systems. Implementing a query that needs multiple joins can be a problem in some cases. Managing consistency between databases is a difficult task.

Every single time a computation is done outside the module boundary, the request has to travel through the network. This results in communication overhead. The Microservices approach will result in slower services.

From a mobile development perspective, a large number of calls to backend Microservices is very expensive in terms of battery usage. It is not possible to build a version of the application for mobile devices only because all calls have to eventually go to the backend servers.

The core idea of Microservices is having a large number of small services, each doing small part of the work. Here, the focus is not only on what such services are doing but on the communications between them. Every single communication between one service and another is a potential place for something that can go wrong. In addition to unit testing, testing a portion of communicating services all together is necessary to obtain a better image of how the system would behave in production.

Knowing what other services expect without hardcoding such requirements may not be an easy task. In contract testing, ingoing and outgoing attributes are checked for conformance with the expected attributes in each case. The development team of each Microservice has to check communication with other services for conformance of contracts.

 ISBN: 978-1-61208-701-6

Sharing code is harder. The objective of Microservices is to create truly independent services. However, if there is a need to share common utility code between services, then the only option is to replicate a functionally across a number of different services.

VII. CHALLENGES

While the concept of Microservices is simple, its implementation is not. The problem is building a system of Microservices. There are no specific rules for many issues: only tradeoffs. One of the underlying tenets of Microservices is that each service runs in its own isolated process. In such case, the question is: how do services find each other to connect? Hence, there is a need for a service discovery mechanism to avoid hard coding the addresses.

One challenge of Microservices is deciding when to include functions inside one service, and when to break them into separate services. The shift to Microservices requires changing the development methodology. The agile approach would be suitable for the Microservices way.

Monitoring the system is another challenge because each service may be running on a different computer or even on a different platform. There is a potential that something might go wrong. Having a mechanism for viewing which service is causing a bottle neck is essential for such systems.

Having dispersed services, and more opened ports leads to a greater attack surface. If each service has its own database, then there is more potential for database related attacks.

Although the Microservices approach offers substantial benefits, a Microservices architecture requires extra machinery, which can impose substantial costs. To enhance the economics of Microservices, it is useful to be integrated with the cloud [12].

Discovery, granularity, and security are among the challenges that faced prior technologies as well, such as Web services [14]. While security and granularity had some solutions, automatic discovery has never been solved.

VIII. CONCLUSION

The term Microservices implies something small, but this name can be misleading since not all services in a Microservices architecture need to be micro [7]. A service will become as big as it needs to be to provide a coherent, efficient, and reliable function. However, it is not about the size. It is about focus and logical cohesion. The main advantage is breaking the system into smaller chunks that can be managed individually.

Before switching to Microservices, a number of questions need to be answered depending on each individual case: why it is needed? Is it scalability? Is it flexibility? Which parts of the business have such needs? An additional issue concerns the readiness of the teams for the journey. If the answers are not clear enough and justified, then finding the correct answer should come first. Unless the goals are clear enough, benefits cannot be measured.

Writing Microservices based application involves many different issues compared to writing monolithic applications. However, transitioning from a monolith is even more difficult than building Microservices from scratch. Advocates for Microservices implicitly suggest that monoliths are outdated. While flexibility and scalability are weak points in monoliths, they may not be priority for all applications.

Everything comes with a cost, and so do Microservices. If the benefits of switching to Microservices do not outweigh the gains, then the decision is not rational. Whether to move the whole monolith to Microservices is a critical question and does not have a black or white answer. Chosen parts of the application can be migrated into Microservices. The Microservices design thinking can be applied to a monolith. Decomposition strategy, and interaction patterns have to be revisited. A monolithic system can still implement asynchronous communication.

REFERENCES

[1] S. Peng, L. Shen , H. Liu and F. Li, "User-Oriented Measurement of Software Flexibility," in *2009 WRI World Congress on Computer Science and Information Engineering*, vol. 7, IEEE, pp. 629-633, 2009,.

[2] M. Elkholy and A. Elfatatry, "Change Taxonomy: A Fine-Grained Classification of Software Change," *IT Professional,* vol. 20, no. 4, pp. 28-36, 2018.

[3] A. Elfatatry, "Dealing with Change: Components Versus Services," *Communications of the ACM,* vol. 50, no. 8, pp. 35-39, August 2007.

[4] P. Jamshidi, C. Pahl, N. Mendonca, and J. Lewis, "Microservices: The Journey So Far and Challenges Ahead," *IEEE Software,* vol. 35, no. 3, pp. 24-35, 2018.

[5] N. Alshuqayran , N. Ali, and R. Evans, "A Systematic Mapping Study in Microservice Architecture," pp. 44-51, 4-6 Nov. 2016.

[6] M. Fowler, "martinfowler.com," 2017. [Online]. Available: https://martinfowler.com/articles/microservices.html. [Accessed January 2109].

[7] J. Thönes, "Microservices," *IEEE Software,* vol. 32, no. 1, pp. 116, 2015.

[8] S. Newman, Building Microservices, CA: O'Reilly Media, Inc., pp. 9-12, 2015.

[9] C. Pautasso, O. Zimmermann, M. Amundsen, J. Lewis, and N. Josuttis, "Microservices in Practice, Part 2: Service Integration and Sustainability," *IEEE Software,* vol. 34, no. 2, pp. 97-104, 2017.

[10] A. Kwan, H.-A. Jacobsen, A. Chan, and S. Samoojh, "Microservices in the modern software world," pp. 297-299, 2016.

[11] S. Green, How To Build Microservices: Top 10 Hacks To Modeling, Integrating & Deploying Microservices, pp. 24-32, 2015.

[12] A. Singleton, "The Economics of Microservices," *IEEE Cloud Computing,* vol. 3, no. 5, pp. 16-20, 2016.

[13] L. S. David, "Practical Use of Microservices in Moving Workloads to the Cloud," *IEEE Cloud Computing,* vol. 3, no. 5, pp. 10-14, 2016.

[14] C. Zeng , Z. Lu , J. Wang , P. Hung, and J. Tian, "Variable Granularity Index on Massive Service Processes,", IEEE 20th International Conference on Web Services, Santa Clara, CA, USA pp. 18-25, 2013.

 ISBN: 978-1-61208-701-6