Monitoring and Analysis of Microservices Performance

Article · May 2017

CITATIONS
9

READS
5,031

1 author:

Saman A. Barakat
University of Duhok
4 PUBLICATIONS **29** CITATIONS

SEE PROFILE

Monitoring and Analysis of Microservices Performance

SAMAN Barakat

University of Duhok, Kurdistan Region, Iraq
Department of Computer Science, College of Science
42001 Duhok, University of Duhok, Iraq, E-Mail1: saman.barakat@uod.ac

Abstract *– Monolithic applications have some drawbacks and issues. In particular, when new features are added to the monolithic applications the code bases become bigger. With Microservices such issues can be avoided. Microservices are software architecture that consists of small, distributed services that work together. It has many advantages such as simplicity, independent service scalability, using different technologies and simplicity in deployment, but requires knowledge in distributed systems. In this research, an existing microservice-based application has been monitored and analyzed. The Kieker framework has been used for monitoring the application performance and the Kieker trace analysis tools have been used to analyze the application.*

Keywords: *monitoring microservices; analysis microservices; microservices architecture.*

I. INTRODUCTION

Developers build different kind of applications based on monolithic architecture. However, when new features are added to monolithic applications, the code bases increase. This makes maintenance very difficult. Developers cannot work independently on monolithic applications, in which all team members have to coordinate all development and re-deployments efforts. The development process becomes more complex over time. Updating the application is difficult because updates require re-deploying the whole application.

Robert C. Martin's definition of the single responsibility principle states "Gather together those things that change for the same reason, and separate those things that change for different reasons." [1]. Microservices take this approach to independent services. Microservices have gained much popularity in industry in the last few years. Microservices communicate with other distributed microservices, as illustrated in Fig. 1. This architecture style has many benefits such as smaller codebases for individual services, ability to update and scale the services in isolation, enabling services to be written in different languages if desired, utilizing varying middleware stacks and even data tiers.

II. CHARACTERISTICS OF MICROSERVICES

The mcroservice architectural style intends to overcome scalability issues of monolithic applications [2].

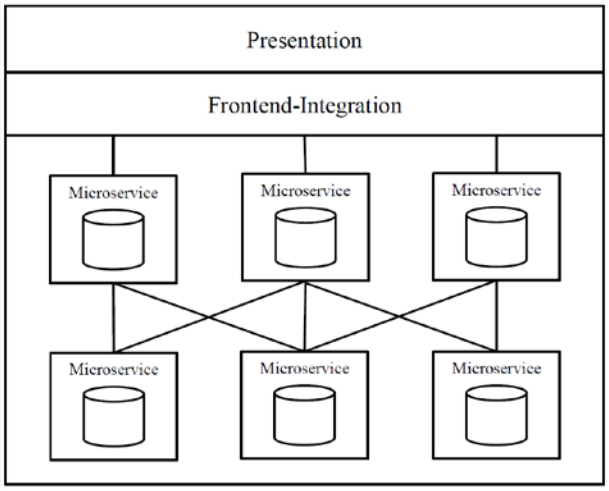

Fig. 1. Microservices architecture.

Microservices have many good characteristics; therefore, many companies have started to build their large applications using this architecture.

Newman [1] lists many advantages of microservices architecture. These benefits can be summarized as follows:

1. Easy to understand: Each service is responsible for only one task; therefore, it requires less code. This means that it is easy to understand and has less risk of changes.
2. Easy to scale: With a large, monolithic application you have to scale everything together. While, with microservices only required services can be scaled.
3. Easy to deploy: With monolithic applications any change in the code requires redeployment of the whole application. This could be a big problem for many organizations with its high degree of risk and disruption. Deploying microservices is much simpler because the scope of a deployment is much smaller.
4. Ability to use a different technology: with microservices, the approach should be to use the best tools and languages for the job, instead of one size fit all.
5. System resilience: If a problem occurred in the monolithic application, the application will stop working, and then many functionalities may not work, While, if one service of the microservices stops working, only a small, particular functionality will be lost.

III. RELATED WORK

Singh [3] has conducted a work on cluster-level logging of containers with containers. He states that gathering and analyzing log information is an essential factor of running production systems to ensure their reliability and to provide important auditing information. Also, he mentioned that collecting the logs of components using containers and Kubernetes is very challenging. Therefore, his research looks at how to overcome these issues.

Amaral et al [4] have done a study on evaluating microservices performance using containers. Containers are used to deploy application either based on monolithic architecture or microservices architecture. Microservices architecture in containers can be achieved by implementing one of two models master-slave or nested containers. The goal of the study was to compare the CPU and the network running benchmark in the two models of microservices architectures.

Villamizar et al [5], conducted research on evaluation of monolithic and microservices architectures using cloud computing. In their study, the Play framework has been used to develop an application based on both, monolithic and microservices architectures. This application was later deployed on a cloud. The performance tests executed on both applications showed the benefits and challenges of implementing microservices applications.

Krylovskiy et al [6] have done a research on design a Smart City IoT platform based on the Microservices architecture. They suggest that this architecture provides significant advantages compared to Service-Oriented Architecture for building.

Alpers et al [7] have worked on a Microservice based tool support for business process modeling. In this article, they have explained how Microservices can be used to build modeling editors and additional services. Also, they have discussed how Microservices can used to perform collaborative modelling techniques, increase reuse of utilized service components and improve their integration into lightweight user interfaces.

Guo et al [8] have proposed a new Cloudware PaaS platform based on Microservice architecture and light weighted container technology. This platform has the characteristics of scalability, auto-deployment, disaster recovery and elastic configuration.

Lacic et al.[9] have introduced a scalable recommender framework. The framework is designed based on the microservices architecture and exploits search technology in order to provide real-time recommendations.

Le et al. [10] have introduced a new architecture solution for the Nevada Research Data Center based on the Microservices architecture. The main purpose of this solution is to provide scalability, reliability, maintainability to the data center.

Patanjali et al. [11] have introduced a design based on Microservices architecture for developing a dynamic rating, charging and billing for cloud service providers. They have focused on the architecture validation using mathematical modeling.

IV. PROBLEM STATEMENT

Microservices architecture has been adapted by many companies to build large scalable applications. Microservices applications consist of many small autonomous services. These services communicate with each other in order to complete some tasks. Since services can fail at any time, detecting the service failure is an important issue. Monitoring microservices can be a key factor to detect service failure earlier. However, till now very few studies have been done on monitoring and analysis of microservice performance. Therefore, this research has been conducted in order to understand how to monitor and analyze microservices performance. The research should be able to answer the following questions:

1. How can microservices applications be monitored?
2. How can microservices performance be analyzed?

V. METHODOLOGY

In order to achieve the aim of this project, an existing Microservice based application will be considered. Then, this application will be monitored using Kieker framework, as suggested by [2]. After the process of monitoring is done, logs are generated. These logs will be analyzed using Kieker's trace analysis tools.

VI. MICROSERVICES IMPLEMENTATION

Microservices can be implemented using many tools and languages, for instance with the Spring, Jolie or Play frameworks.

A. JOLIE programming language

Jolie is an open-source programming language used to build microservices based applications. In the Jolie programming, each program is considered as a service that can communicate with other programs using a network for sending and receiving messages. Jolie supports an abstraction layer that allows services to communicate using different methods such as TCP/IP sockets and local in-memory communications between processes.

B. SPRING framework

The Spring framework can be used to implement microservices applications. It is an application framework and inversion of control container for the Java platform. The framework's core features can be used by any Java application, but there are extensions for building web applications on top of the Java EE platform.

C. PLAY framework

Play framework is an open source web application framework, written in Scala and Java. This framework

follows model–view–controller (mvc) architectural pattern. It aims to optimize developer productivity. This framework is used by (Mario et al) to implement and evaluate Microservice based application [5].

For the purpose of this research, an existing microservice-based application has been used [12]. This application has been monitored and analyzed using Kieker analysis tools. The application consists of three services which are Web-Service, Account-Service and Registration-Service as can be seen in Fig. 2.

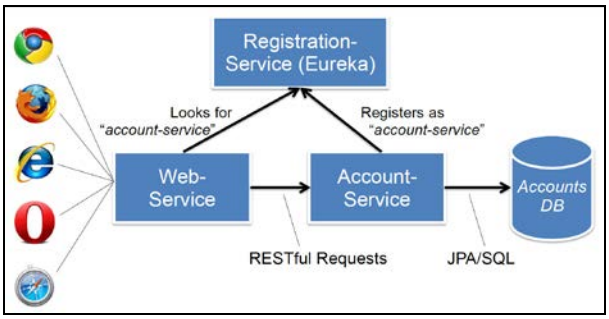

Fig. 2. Microservices-based application.

VII. KIEKER FRAMEWORK

Kieker is a framework developed in Java under the Apache License Version 2.0 used for monitoring and analyzing software performance [13], [14]. Kieker's Architecture consists of two main components: Kieker.Monitoring and Kieker.Analysis. The Kieker.Monitoring component is responsible for program instrumentation, data collection, and logging. It uses measurement probes for the instrumentation of software systems and writers to collect and store the observed data. The other component is Kieker.Analysis, which is used to read the monitored data and analyze and visualize it. The SPEC research group has accepted and published the Kieker framework as a recommended tool for quantitative system evaluation and analysis as part of SPEC research group's tool repository [15].

VIII. MONITORING

The microservice application has been monitored using the Kieker framework. The monitored data is produced and saved in a folder. This folder contains two types of files which are a map file and monitored data. The map file shows what type of operations have been used. The second file has several columns such as type of operation, the method that has been invoked, start time and end time.

IX. ANALYSIS

Kieker framework can be used to analyze the software performance. Kieker has trace analysis tools, in which it uses monitored logs to analyze and visualize the application performance. The tool depends on two third-parties programs which are GraphViz and GNUPlotzunit. It produces several types of diagrams such as sequence diagrams, call trees and dependency graphs. In this paper, a microservice application has been analyzed and visualized using Kieker's trace analysis tools.

X. RESULTS

An existing microservice application, developed using the Spring framework, has been monitored and analyzed as a starting. The research has used the Kieker framework for monitoring microservices performance during run time and Kieker's trace analysis tools has been used for analysis of the application.

As a result, the Fig. 3 shows the deployment operation dependency graph with the operation executing time in "Account-Service" and the Fig. 4 shows the deployment component dependency graph in the "Account-Service".

In the deployment operation dependency graph as can be seen in Fig. 3, it shows the performance of each method in the microservice application. The graph shows how many times each method has been visited. It shows the minimum, average, maximum and total execution time in each method.

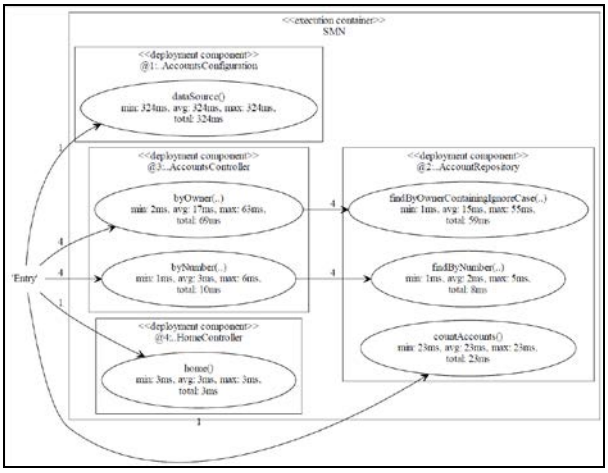

Fig. 3. Deployment operation dependency graph.

In the deployment component dependency graph as can be seen in Figure 4, shows the performance of each component in the Microservices applications. It shows how many times each component has been visited. Also, it shows the minimum, average, maximum and total execution time in each component.

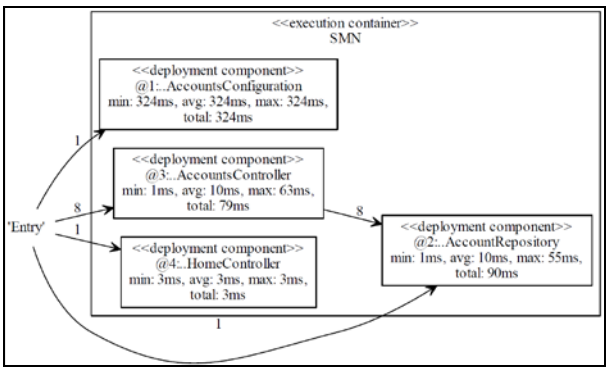

Fig. 4. Deployment component dependency graph

XI. CONCLUSIONS AND FUTURE WORK

The aim of this ongoing research project is monitoring and analysis of microservices performance. There are many frameworks and tools that can be used to implement applications based on microservices architecture for instance the Play and the Spring frameworks. In order to achieve the aim of the project, an existing microservice application, developed using the Spring framework, has been monitored and analyzed. The research has used the Kieker framework for monitoring microservices performance during run time and Kieker's trace analysis tools has been used for analysis of the application.

Future work aims at extending this analysis with ExplorViz which provides software landscape visualization [16] and 3D application visualization [17]. Microservices-based applications are structured into independently deployable, distributed self-contained systems. Such architectures blur the boundaries of landscapes and applications. The new (commercial) monitoring tool Instana for instance, is specifically designed for monitoring such microservices-based systems. Instana provides a 3D visualization on the landscape level. It would be interesting to investigate how 3D visualizations could be beneficial on ExplorViz' landscape level, in combination with the 3D visualization on the application level.

ACKNOWLEDGEMENT

Thanks to Wilhelm Hasselbring for feedback on earlier versions of this paper.

REFERENCES

[1] S. Newman, *Building Microservices*. O'Reilly Media, Inc., 2015.

[2] W. Hasselbring, "Microservices for Scalability," in *International Conference on Performance Engineering (ICPE 2016)*, 2016, pp. 133–134.

[3] S. Singh, "Cluster-level Logging of Containers with Containers," *Queue*, vol. 14, no. 3, pp. 30:83–30:106, May 2016.

[4] M. Amaral, J. Polo, D. Carrera, I. Mohomed, M. Unuvar, and M. Steinder, "Performance Evaluation of Microservices Architectures using Containers," in *14th IEEE International Symposium on Network Computing and Applications*, 2015, pp. 27–34.

[5] M. Villamizar, O. Garcés, H. Castro, M. Verano, L. Salamanca, R. Casallas, and S. Gil, "Evaluating the monolithic and the microservice architecture pattern to deploy web applications in the cloud," in *Computing Colombian Conference (10CCC), 2015 10th*, 2015, pp. 583–590.

[6] A. Krylovskiy, M. Jahn, and E. Patti, "Designing a Smart City Internet of Things Platform with Microservice Architecture," *Proc. - 2015 Int. Conf. Futur. Internet Things Cloud, FiCloud 2015 2015 Int. Conf. Open Big Data, OBD 2015*, pp. 25–30, 2015.

[7] S. Alpers, C. Becker, A. Oberweis, and T. Schuster, "Microservice Based Tool Support for Business Process Modelling," *2015 IEEE 19th Int. Enterp. Distrib. Object Comput. Work.*, pp. 71–78, 2015.

[8] D. Guo, W. Wang, G. Zeng, and Z. Wei, "Microservices Architecture Based Cloudware Deployment Platform for Service Computing," *2016 IEEE Symp. Serv. Syst. Eng.*, pp. 358–363, 2016.

[9] E. Lacic, E. Lex, and D. Kowald, "ScaR : Towards a Real-Time Recommender Framework Following the Microservices Architecture," no. October, 2015.

[10] V. D. Le, M. M. Neff, R. V. Stewart, R. Kelley, E. Fritzinger, S. M. Dascalu, and F. C. Harris, "Microservice-based architecture for the NRDC," *Proceeding - 2015 IEEE Int. Conf. Ind. Informatics, INDIN 2015*, pp. 1659–1664, 2015.

[11] S. Patanjali, B. Truninger, P. Harsh, and T. M. Bohnert, "CYCLOPS: A micro service based approach for dynamic rating, charging & billing for cloud," *Proc. 13th Int. Conf. Telecommun. ConTEL 2015*, 2015.

[12] P. Chapman, "Microservices with Spring," *Spring blog*. 2016.

[13] A. van Hoorn, M. Rohr, W. Hasselbring, J. Waller, J. Ehlers, S. Frey, and D. Kieselhorst, "Continuous Monitoring of Software Services: {D}esign and Application of the {K}ieker Framework," Nov. 2009.

[14] A. van Hoorn, J. Waller, and W. Hasselbring, "Kieker: {A} Framework for Application Performance Monitoring and Dynamic Software Analysis," in *Proceedings of the 3rd ACM/SPEC International Conference on Performance Engineering (ICPE~2012)*, 2012, pp. 247–248.

[15] 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, 2015.

[16] F. Fittkau, S. Roth, and W. Hasselbring, "{ExplorViz}: Visual Runtime Behavior Analysis of Enterprise Application Landscapes," in *23rd European Conference on Information Systems (ECIS 2015 Completed Research Papers)*, 2015, pp. 1–13.

[17] F. Fittkau, J. Waller, C. Wulf, and W. Hasselbring, "Live Trace Visualization for Comprehending Large Software Landscapes: The {ExplorViz} Approach," in *1st IEEE International Working Conference on Software Visualization (VISSOFT 2013)*, 2013, pp. 1–4.