International Journal of Scientific Research and Engineering Development-– Volume 4 Issue 3, May -June 2021

Available at www.ijsred.com

RESEARCH ARTICLE OPEN ACCESS

Performance Monitoring for Microservices Architecture Based Applications

Mohammad Hanan Bhat*, Dr Anala MR**

*(Student at Department of Information Science Engineering, RV College of Engineering, Bengaluru
Email: mohammadhb.is17@rvce.edu.in)

** (Faculty at of Information Science Engineering, RV College of Engineering, Bengaluru
Email:analamr@rvce.edu.in)

----------**********----------

Abstract:

Microservices based architectural style of application development is quite famous in industries nowadays, because it complements the DevOps principles. Microservices are developed rapidly which can lead to serious performance degradation in some cases, especially where unit and integration tests are ignored. It is observed that performance monitoring is often considered as a tedious and unnecessary task in software development lifecycle. Performance is surely a key non-functional requirement but still there is very little attention paid towards it. This paper focuses on the need of performance monitoring and outlines the roadmap for performance testing of microservices with the aim of moving from Test-Driven Development to Performance-Driven Development.

Keywords ***—Performance Evaluation Tool, Performance Oriented Development, Microservices Architecture, Endurance Benchmarking, Load Generator.***

----------**********----------

I. INTRODUCTION

Microservices is the most prominent architectural design for developing lightweight standalone applications serving unique functionalities. With the advent of cloud computing technology and the pay-per-use model offered by the cloud computing service providers, standalone applications with fine-granular design and loosely coupled services have gained huge popularity. Microservices allows developers to develop applications in cloud-native environments with fine-grained control and hence they became extensively used in the industries.

Since microservice applications serve a unique functionality and are independent of other services, they can be developed much faster in an industry when compared to traditional software. Traditional software applications can be divided into multiple autonomous components which can be developed, deployed, and maintained independently without any influence of other services, also known as microservices. These microservices communicate with each other through a well-defined interface to realise the overall business functionality. Compared to traditional monolithic software architecture, microservice-based architecture leads to a higher frequency and pace of application releases, encouraging rapid deployments from development to production. Contrary to monolithic architecture, microservices support heterogeneity in technology i.e., different services in a system can use different technologies to realise the business value and performance goals[1]. Due to this, there is a stark difference in conducting the performance testing for a microserviceapplication over a traditional software application.

Microservices are highly available, flexible, scalable, isolated, and secure. these qualities make them suitable for deploying in the cloud and exploiting the benefits of cloud environments.

microservices are generally containerized and hosted in a cloud-nativeenvironment, therefore the actual performance of the application could significantly differ, depending on the Virtual Machine Instance (VMI) and the application running[2]. The usage of resources by the microservices varies based on the implemented functionality and the workload [3]. The container resource specification file specifies the CPU, memory and scaling constraints assigned to the containers. However, this specification is dynamic in nature and is managed by a container orchestration tool such as Kubernetes, Docker Compose, Swarm, Apache Mesos etc. This key difference in the resource utilisation of cloud applications leads to the inability to develop a uniform performance monitoring system for microservice-based applications[2].

There has been considerable research done in the field of performance engineering in the case of traditional software systems, but very little attention has been paid to performance engineering for microservices despite performance being the absolute necessity to achieve flexibility, scalability, and availability of microservices [4]. In the next section, the need for performance monitoring in microservices and the importance of moving from Test-Driven Development (TDD) to Performance-Driven Development (PDD) will be discussed.

II. NEED FOR PERFORMANCE TESTING

Performance testing is a kind of software testing that is aimed at identifying the bottlenecks in a system. It is strictly attributed to non-functional testing which focuses only on how the application behaves under a specific workload. Very often performance testing is neglected till the end of the software development life cycle as the project winds down. This results in a software that does not comply with Service Level Objectives (SLO).

With microservices architecture, many services interact with each other and undergo configuration changes rapidly. Performance degradation in any service can lead to bottlenecks and hence, failure of an entire application. Therefore, it is critical for businesses to invest an appropriate amount of time and effort in performance testing [5].

Following points emphasise the importance of performance testing in microservice applications:

- *Capacity Planning* - Microservice capacity is identified as the maximum number of requests a service can take before violating SLO [3]. Stress testing helps in identifying the individual capacities of microservices and hence, helps in the overall capacity planning for applications. Microservices allow us to increase the capacity of a system by increasing the replicas of only those services which are underperforming during higher workloads which leads to effective resource utilisation and cost-savings.
- *Identifying Bottlenecks* - Load testing helps to understand the behaviour of the services and various communication interfaces between these microservices. A bottleneck occurs when there is a halting of data between these services. Performance testing provides diagnostic information to eliminate the bottlenecks [6].
- *Resolving Scalability Issues* - Performance testing can unfold resource scarcity related issues. If the application is not able to handle the incoming requests, it can lead to slower response and high error rates. Identifying the right sizing of resources to each microservice is therefore quite essential to meet SLO.
- *Software Configuration Issues* - Since microservices keep evolving at a very high pace, it becomes absolutely necessary to verify if the service is meeting the SLO requirements. Unlike unit testing which aims to verify the functional correctness of the code, the goal of load testing is to log key performance-related metrics of a service.

- *Bug Tracing in non-production environments* - In order to improve the user experience and reduce the number of bugs found in production, it is important to fully test the integrated application end to end. Oftentimes developers do not put enough effort into performance testing which can lead to catastrophic failures in production. Therefore, it is extremely important to simulate the required workload, in order to reduce bugs in production.

In many cases, performance testing is undertaken by a separate team of testers other than the developers. This strategy can often lead to partial testing of the system as the performance testing requires breaking into the system and covering all the possible use cases[7]. Therefore, it should be encouraged that the development team should also perform load testing and analyse the performance metrics, thus moving towards performance-driven development.

III. TYPES OF PERFORMANCE TESTING

Performance testing helps us to expose the underperforming services which are not optimised for scalability. As a result, it prevents application crashes due to increased workload in the production environment. To model the different workload conditions and analyse the performance of services, different types of performance testing are conducted. Since microservices are continuously evolving, it becomes necessary to create a baseline and compare the performance of services after code changes or updates in hardware or software components. It should be done to ensure that the performance of the system does not downgrade with time.

Few types of performance testing for microservices are as follows:

- *Load Testing* - This type of testing is used to analyse the performance of a system under some expected user loads. It is generally done for normal workload conditions, termed as representative load testing. It helps in the capacity planning of microservices.
- *Stress Testing* - Stress Testing aims at determining the performance of the system at loads that are beyond the normal working conditions mentioned in SLO. It allows testing of the system for sudden and unexpected workload for a shorter period of time. It helps to set the scaling configuration settings to accommodate the sudden bursts of load.
- *Soak Testing* - Soak Testing or Endurance Testing allows the developer to understand how the system would handle a normal workload over an extended period of time. It is important to note that soak testing does not entertain scaling of services. It aims at handling a constant load with static resources.
- *Spike Testing* - This is a type of stress testing that tests the system's reaction to a sudden and rapid increase in user load for repeated and short duration of time. The load generated by users is higher than normal working conditions as in case of stress testing.

IV. ROADMAP FOR PERFORMANCE TESTING

To enforce the principles of performance-driven development in the software development life cycle, it is important to have a standard roadmap for testing microservices. This simplifies the process of testing these services and ensures quicker delivery of results. Following are the steps required to conduct the performance testing:

i. *Set up the environment for testing* - The environment for testing should resemble the production environment so that there is no disparity between the performance of

service in these environments. Microservices are developed using DevOps and it should be the responsibility of developers to manage the infrastructure of the testing environment.

ii. *Identify the right load generator* - Load is defined as the sequence of requests or messages that are sent to the microservices. A load generator is a tool consisting of hardware or software components that provide this load to the communication interface of microservices. Enhancing the performance of microservice often demands the use of load generators which mimic the actual workload. A representative load test uses workload characteristics according to the user behaviour in production [8]. The requirements for such a load generator are: simulating load, automation of workflow, integration with CI/CD tools and providing better insights [9].

iii. *Identify key metrics and monitoring tools* - Metrics are important to identify the quality and effectiveness of performance testing. These metrics are generally part of SLO as performance testing is used to verify and configure the service level agreements. Few of the general metrics used to monitor the microservices are: CPU utilisation, memory used, disk utilisation, throughput, DB calls made per minute from a service, error rate, number of transactions passed/failed, average response time, wait time etc.

To monitor the performance metrics, it is necessary to find the right tool as these tests can go for an extended period of time and hence, becomes inconvenient to monitor. Some of the tools available are:

- JMeter
- Grafana
- Influx DB
- AWS CloudWatch
- AppDynamics
- AutoPerf[10]

iv. *Interpret the results and make corrective action* - The metrics recorded from the performance testing are used to create a baseline for future comparisons. Therefore it is necessary to derive all the attributes from these metrics and visualise them. Based on the analysis of the results, suitable corrective actions have to be taken to meet the service level objectives. This generally involves infrastructure changes and scaling of resources.

V. SUMMARY

Microservices is a unique approach to software development which is famous for developing lightweight standalone applications which interact together to fulfil the business goal. Microservices undergo rapid configuration changes in and hence, to avoid bottlenecks and failure, performance testing must be implemented. Traditional monolithic systems often undergo function testing followed by non-functional performance testing. This approach does not work for microservices since these applications use multiple services to communicate and all of them might not be production ready. Capacity planning, identification of bottlenecks, bug tracing in non-production environments, resolving of scalability and software configuration issues are few notable advantages of performance testing in microservices. There are several types of performance testing for microservices. A few of them include load testing, stress testing, soak testing and spike testing, each having their own objective and importance. A standard roadmap for testing microservices ensures that the principles of performance-driven development are accommodated in the software development life cycle.

ACKNOWLEDGEMENTS

We would like to thank Dr B.M. Sagar, Professor & HOD, Department of Information Science Engineering, RV College of Engineering, who constantly encourages students to indulge in

research-based activities and this paper is an outcome of his efforts.

REFERENCES

[1] O. Al-Debagy and P. Martinek, "A comparative review of microservices and monolithic architectures," in *IEEE 18th International Symposium on Computational Intelligence and Informatics (CINTI)*, Budapest, Hungary, 2018.

[2] F. Samreen, Y. Elkhatib, M. Rowe and G. S. Blair, "Daleel: Simplifying cloud instance selection using machine learning," in *NOMS 2016-2016 IEEE/IFIP Network Operations and Management Symposium*, 2016, Istanbul, Turkey.

[3] A. Jindal, V. Podolskiy and M. Gerndt, "Performance Modeling for Cloud Microservice Applications," in *Proceedings of the 2019 ACM/SPEC International Conference on Performance Engineering*, Mumbai, India, 2019.

[4] R. Heinrich, A. K. H. van Hoorn, F. Li, L. E. Lwakatare, C. Pahl, S. Schulte and J. Wettinger, "Performance Engineering for Microservices: Research Challenges and Directions," in *Proceedings of the 8th ACM/SPEC on International Conference on Performance Engineering Companion*, L'Aquila, Italy, 2017.

[5] C. M. Aderaldo, N. C. Mendon, C. Pahl and P. Jamshidi, "Benchmark requirements for microservices architecture research," in *2017 IEEE/ACM 1st International Workshop on Establishing the Community-Wide Infrastructure for Architecture-Based Software Engineering (ECASE)*, 2017.

[6] J. Cong and B. E. Wolfinger, "A Unified Load Generator Based on Formal Load Specification and Load Transformation," in *Proceedings of the 1st international conference on Performance evaluation methodolgies and tools*, Pisa, Italy, 2006.

[7] N. Athanasiades, R. Abler, J. Levine, H. Owen and G. Riley, "Intrusion detection testing and benchmarking methodologies," in *First IEEE International Workshop on Information Assurance, 2003. IWIAS 2003. Proceedings.*, 2003.

[8] H. Schulz, T. Angerstein, D. Okanovi'c and A. van Hoorn, "Microservice-tailored generation of session-based workload models for representative load testing," in *2019 IEEE 27th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*, 2019.

[9] A. De Camargo, I. Salvadori, R. d. S. Mello and F. Siqueira, "An architecture to automate performance tests on microservices," in *Proceedings of the 18th International Conference on Information Integration and Web-based Applications and Services*, Singapore, 2016.

[10] S. S. Shirodkar and V. Apte, "AutoPerf: an automated load generator and performance measurement tool for multi-tier software systems," in *Proceedings of the 16th international conference on World Wide Web*, Banff, Alberta, Canada, 2007.

