Microservice architecture design for autograder using distributed architecture

ECS
240th ECS Meeting
Digital Meeting, Oct 10-14, 2021
We are going fully digital!
Attendees register for free!
REGISTER NOW

Microservice architecture design for autograder using distributed architecture

R Elsen*, M R Nashrulloh, R Cahyana, A Mulyani and A Latifah

Department of Informatics, Sekolah Tinggi Teknologi Garut, Jalan Mayor Syamsu 1, Garut 44151, Indonesia

*rickardelsen@sttgarut.ac.id

Abstract. An autograder is a system for grading student code submission so the lecturer and student can get the result immediately after code has been submitted. This system using distributed architecture which uses multiple backends as a grader in purpose to reduce execution time caused by the queue. This system needs some components to make it work. Based on distributed architecture applied in this system design, the components need to be separated and run independently but still running their respective orchestras. In this paper, we propose a microservice architecture design to determine components that might be needed by autograder to complete its purpose and run well.

1. Introduction

Autograder is used to grade student submission based on execution results and compare it with some test cases so the lecturer can know student's grades immediately after this process is done [1]. In a monolithic architecture, all this routine mostly stuck on code execution and it creates more delay between request and result because another request must be waiting in the queue until the previous request finished to grade. But in distributed architecture [2], it used more than one backend to execute code and grade the result so it will shorten queue and waiting time.

One of the applications of distributed architecture is microservice. Microservice allows components to run independently and minimize coupling between components [3]. Microservice also makes it easier to maintain and deploy each component with a low impact on another component [4]. In autograder, microservice can be used to separate the backend with other components like user interface or database. Load balancer will also be an independent component for orchestrating grader backends. By using microservice, autograder can deploy more backends independently as needed so it can adapt to the number of requests based on the number of users or the severity of the task assigned, and also adapt to the different programming language used in the task. In the future, if autograder add more feature, it just needs to deploy new component and modify some components so another component that does not require changes will not be impacted.

2. Methodology

We use observation methodology by observing some information about microservice gathered from websites and papers [5]. As a result, we got the technique to design microservice for autograder. We observe four topics as basic knowledge of microservice architecture:

- We observe best practices in microservice so we know about what important thing in microservice [6,7].
- Since autograder use monolithic architecture, we observed about how to migrate from a monolithic application to microservice [8,9].
- We observe about quality attributes in microservice architecture [10,11].
- Since we try to migrate from monolithic to microservice, we observe about how to build an application using microservice [12-15].

3. Result and discussion

3.1. Distributed architecture of Autograder

In this architecture, autograder use multiple backends to execute codes and grade results [2]. All backends managed by load balancer for orchestrating requests to prevent colliding between requests. It applying multiple backends so it can execute more than one code at a time and reduce delay caused by the long queue.

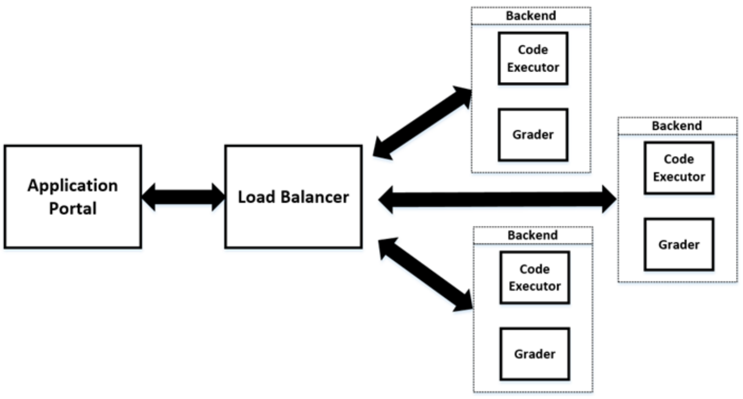

Figure 1. Autograder with distributed architecture [2].

As seen in Figure 1, multiple backends that contain code executor and grader work behind the load balancer. This load balancer will determine which backend will receive the request based on its internal queueing system. It also manages all grading results and sends the result to the application portal so the lecturer and student can know the grades.

3.2. Autograder components

Based on Figure 1, there are four main components of autograder:

- Application portal. This component is used by the lecturer and student. The lecturer can create courses and give assignments to students. Students can send submission based on assignments. Both can see the grades in this portal.
- Load balancer. This component responsibility is orchestrating requests from the application portal and distribute it to all available backends. It must describe how requests will be queued, how to choose idle backend, and how to manage responses from multiple backends.
- Code executor. This component is used for executing the student’s submission. It will execute the submission using some input test cases that already determined by the lecturer while making the assignment.
- Grader. This component will compare execution results with output test cases. It also grades student’s submission based on the value determined by the lecturer for every test case passed.

Based on four autograder main components, we determine seven components needed by autograder to run properly:

- User interface. This component will be used for showing all features to lecturers and students.

- Application backend. This component needed as an engine for user interface to make it work properly.
- Database. This component will store all data related to autograder.
- Volume. This component will store all files related to autograder.
- Load balancer. This component will orchestrate backends and manage all requests and responses.
- Code executor. This component will execute student submission with the input test cases.
- Grader. This component will grade the execution result by comparing it with output test cases.

3.3. Microservice architecture of Autograder

Based on autograder components, we propose a design for autograder using microservice architecture:

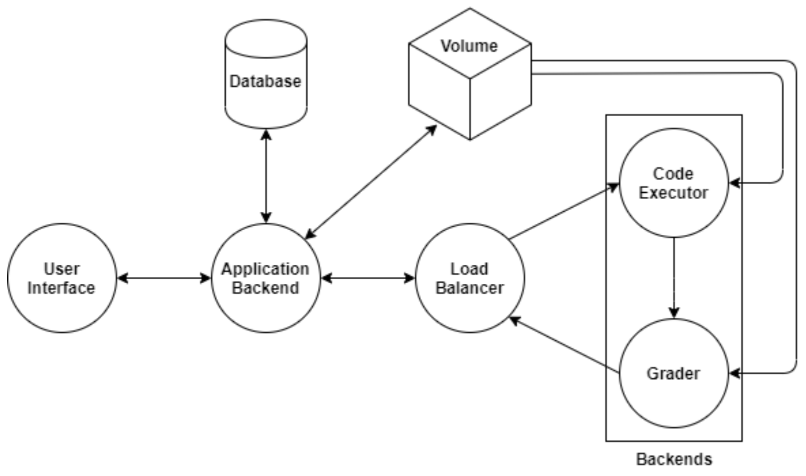

Figure 2. Microservice architecture design of Autograder.

As seen in Figure 2, the user interface will communicate with the application backend to send user requests or showing data needed by users. Application backend will read or save data to the database like user data, assignment data, and course data. When students send a submission, the application backend will store submission data to the database and save submission files in the volume. Application backend also communicates with the load balancer by sending information that a new request is ready. Load balancer will manage the queue of requests since there will be more than one student send submission at the same time. Load balancer also manages to which backend the request will be sent based on the idle status of backend. If all backends are not idle, load balancer will hold requests to stay in queue until there is an idle backend ready to grade. When backend ready to grade, load balancer will give information to code executor about which submission must be executed. Load balancer also gives information about which test case must be included while doing execution. Code executor will load submission and test cases from volume and execute code until it shows results. Code executor will send information about test cases to grader along with execution result so grader can load the right output test cases from volume. Grader will compare execution result with output test case and determined value per test case passed by the lecturer. After the grader finishes grading, the result will be sent to load balancer and load balancer will forward it to application backend and show the result in user interface. Note that autograder can use more than one backend in a distributed architecture. But in microservice, components can run independently so it described as one component in the design.

We have done a preliminary test to see the difference by using microservice. With the same number of backends, the result shows it not affect performance. But the main purpose of using microservice is to make it easier to deploy new components, like backend or new feature. Without microservice, we need to shut down all components before adding more backend. But with microservice, we just need to deploy some backend and register these new backends to load balancer.

4. Conclusion

In microservice architecture, all components can run independently. Autograder backend must be separated and multiplied so using microservice will support its purpose to grade student submission using multiple backends. The main key of fast grading is the queueing method in the load balancer and orchestrating multiple backends based on request and queue. By using microservice, autograder can add more backend and feature without lost its service. This proposed architecture is based on best practices. Testing is only limited to show it can deploy new components without impacting all autograder service. Perhaps in the future, it will be applied and tested in autograder.

Acknowledgment
The Authors acknowledge Sekolah Tinggi Teknologi Garut for supporting and funding this paper.

References

[1] Danutama K and Liem I 2013 Scalable Autograder and LMS Integration *Procedia Technology* **11** 388-39
[2] Elsen R, Latifah A and Sutedi A 2019 Distributed architecture for autograding system *Journal of Physics: Conference Series* **1402** 077017
[3] Richardson C 2020 *What are microservices?*
[4] Huston T 2020 *What are Microservices?* (SmartBear)
[5] Dudovskiy J 2020 Observation *Research-Methodology*
[6] Weeks D 2020 *10 Best Practices for Microservice Architectures* (DZone Microservices)
[7] Kamaruzzaman M 2020 *Effective Microservices: 10 Best Practices* (Medium)
[8] Ponce F, Marquez G adn Astudillo H 2019 Migrating from monolithic architecture to microservices: A Rapid Review *38th International Conference of the Chilean Computer Science Society (SCCC)*
[9] Mazlami G, Cito J and Leitner P 2017 Extraction of Microservices from Monolithic Software Architectures *IEEE International Conference on Web Services (ICWS)*
[10] Li S 2017 Understanding Quality Attributes in Microservice Architecture *24th Asia-Pacific Software Engineering Conference Workshops (APSECW)*
[11] Cojocaru M, Uta A and Oprescu A 2019 Attributes Assessing the Quality of Microservices Automatically Decomposed from Monolithic Applications *18th International Symposium on Parallel and Distributed Computing (ISPDC)*
[12] Fetzer C 2016 Building Critical Applications Using Microservices *IEEE Security & Privacy* **14**(6) 86-89
[13] Wagner J 2020 *Building an Application: Strategies for Microservices* (Swagger)
[14] Oparin G, Bogdanova V, Pashinin A and Gorsky S 2019 Microservice-oriented Approach to Automation of Distributed Scientific Computations *42nd International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO*
[15] Rademacher F, Sorgalla J and Sachweh S Challenges of Domain-Driven Microservice Design: A Model-Driven Perspective *IEEE Software* **35**(3) 36-43