Towards coverage criteria for serverless applications

Stefan Winzinger

Distributed System Group, University of Bamberg, Germany
stefan.winzinger@uni-bamberg.de

Abstract. Serverless computing is a popular execution model where a cloud provider is responsible for the allocation of resources. Stateless serverless functions build the basis for a serverless application. The statelessness of the functions eases the scalability of the system. A serverless application is built by connecting the serverless functions with other services. Even though the behavior of the serverless functions is easy to test in isolation, the behavior emerging by their integration is hard to predict. Therefore, integration tests are necessary. However, there are no coverage criteria available yet for this specific class of applications making test case sets comparable. Therefore, we introduce characteristics and approaches which can be used for coverage criteria.

Keywords: Serverless Application, FaaS, Integration Testing, Coverage Criteria

1 Introduction

Serverless computing is a popular computing model which is supported by the established IT companies ([1], [3], [4], [2]). But there are also open source solutions ([6], [5]) available.

Stateless serverless functions are used in a serverless application. The cloud provider can easily scale them up if the workload increases. This is done by the usage of stateless, mostly short-running functions whose degree of resource concurrency can be adjusted automatically on function level [7]. By connecting them with each other or other resources, like instances of databases, a complex application is created. However, the behavior emerging in such a complex system is hard to test if the components are only tested in isolation. Making a platform provider responsible for the execution of the serverless resources makes it even harder to cover all relevant scenarios. Therefore, integration tests are needed testing the behavior of resources used together.

However, it is difficult to decide if a test case set is good enough or if new test cases could improve the test quality if there is no possibility to compare test cases. Therefore, coverage criteria are needed indicating the degree of coverage and thus making them comparable. But no coverage criteria are available yet focusing on the coverage of the specific class of serverless applications. Many coverage criteria are available in literature (e.g., [8], [9]) whereas there are no criteria available yet focusing on serverless applications. Therefore, we plan to introduce coverage approaches for specific serverless characteristics.

2 Approaches for Coverage Criteria

In the following, we discuss characteristics of serverless applications and coverage concepts which seem reasonable to be used for the creation of coverage criteria.

Services: The usage of different services, like serverless functions, databases or interfaces, can be used as a coverage criterion by indicating the percentage of called resources of these services. This would enforce the confidence that the resources of the services are deployed correctly and work properly. A criterion subsuming this one is the coverage of all connections between the resources. This ensures that the communication between the resources works correctly, e.g., that the used data format is interpreted correctly.

Parallelism: Because of the parallelism of serverless applications resulting from the scalability of the serverless functions, race conditions can occur if the same data are accessed by at least two different workflows where at least one workflow writes data. Coverage criteria checking these problems could evaluate if hot spots of these conflicts are called in certain situations by the test case sets. Both the parallel call of the same workflow and the parallel call of different workflows have to be checked to fulfill a coverage criterion.

Execution time: The execution time can vary depending on the hardware which is assigned to a function. Functions are sometimes executed with more hardware resources than what is assigned to them in their settings. This results in a faster execution of the function. Therefore, different execution times have to be considered if a function is tested.

Functions which aren't used for a while take longer to be executed which is called "cold start". By testing the application with different cold start delays, potential effects of the cold start can be covered.

Patterns using the time have to be defined and used as coverage criteria. Thus, different kinds of errors can be revealed. Different execution times can influence the workflow which can even provoke race conditions. But also errors caused by a time out are possible.

Access rights: Access rights assigned to serverless functions can also be used as coverage criteria indicating the usage of the rights. Test cases must use each of the assigned access rights. If no test cases can be created for the relevant rights assigned, the necessity of the assigned access right has to be reconsidered.

Dataflow: Since the functions are stateless in a serverless application, their states have to be persisted somewhere else. This is either in a data storage or within the data passed from one resource to another. By applying well-known coverage criteria like the coverage of certain code between the definition and the use of a variable ([10]) and transferring them to a serverless application, test case quality could be improved.

Workflow: By just considering the resources of the application and its order of execution, different workflows are identified. These workflows can also be used for the creation of coverage criteria by transferring known coverage criteria to serverless applications. The coverage criteria to cover all paths suggested by [9]) could be transferred to a coverage criteria requiring all workflows to be covered.

Errors: There are some errors which are specific for serverless applications whose occurrence can be used for coverage criteria. Depending on the configuration and the platform provider, each function has limited resources like memory and CPU power which was assigned to it. Therefore, if the execution of a function takes too much time or memory, the execution might throw an error and be restarted. Therefore, test cases are needed covering these errors and their recovery scenarios.

3 Conclusion and Outlook

In this paper, we suggested how coverage criteria for a serverless application can be created by focusing on serverless-specific characteristics and applying known concepts for these systems. Depending on the criteria used, different classes of errors can be better revealed and prevented by testing.

In our future work, we plan to create concrete coverage criteria for serverless applications and to automate the test case generation by using a model describing the serverless application. These test cases shall be optimized to the fulfillment of serverless-specific coverage criteria. Furthermore, we want to support the tool support to track these criteria. Thus, the quality of this specific class of complex applications can be improved.

References

1. AWS Lambda (accessed January 9, 2019), `https://aws.amazon.com/lambda/`
2. Azure Functions (accessed January 9, 2019), `https://azure.microsoft.com/en-us/services/functions/`
3. Google Cloud Functions (accessed January 9, 2019), `https://cloud.google.com/functions/`
4. IBM Cloud Functions (accessed January 9, 2019), `https://www.ibm.com/cloud/functions/`
5. OpenFaaS (accessed January 9, 2019), `https://github.com/openfaas`
6. OpenLambda (accessed January 9, 2019), `https://github.com/open-lambda`
7. Baldini, I., Castro, P., Chang, K., Cheng, P., Fink, S., Ishakian, V., Mitchell, N., Muthusamy, V., Rabbah, R., Slominski, A., Suter, P.: Serverless computing: Current trends and open problems. In: Research Advances in Cloud Computing, pp. 1–20. Springer Singapore (2017)

8. Frankl, P., Weyuker, E.: An applicable family of data flow testing criteria. IEEE Transactions on Software Engineering 14(10), 1483–1498 (oct 1988)
9. Huang, J.C.: An approach to program testing. ACM Computing Surveys 7(3), 113–128 (sep 1975)
10. Rapps, S., Weyuker, E.: Selecting software test data using data flow information. IEEE Transactions on Software Engineering SE-11(4), 367–375 (apr 1985)