Efficient test execution in End to End testing

Resource optimization in End to End testing through a smart resource characterization and orchestration

Cristian Augusto†
Department of Computing
University of Oviedo
Gijon ASTURIAS SPAIN
augustocristian@uniovi.es

ABSTRACT

Virtualization and containerization have been two disruptive technologies in the past few years. Both technologies have allowed isolating the applications with fewer resources and have impacted fields as Software Testing. In the field of testing, the execution of the containerized/virtualized test suite has achieved great savings, but when the complexity or the cost of deployment arises, there are open challenges like the efficient execution of End to End (E2E) test suites. This paper proposes a research problem and a feasible solution that looks for improving resource usage into the E2E tests, towards smart resource identification and a proper organization of its execution in order to achieve efficient and effective resource usage. The resources are characterized by a series of attributes that provide information about the resource and to its usage during the E2E testing phase. The test cases are grouped and scheduled with the resources (i.e. deployed in parallel in the same machine or executed in a fixed arrange), in order to make an efficient execution of the entire test suite, reducing its total cost/time

CCS CONCEPTS

Software verification and validation

KEYWORDS

Orchestration, End to End Testing, Efficient use of resources, Containerization

1 Introduction

Over half a century, virtualization has achieved important savings in resource terms and has preceded the modern containerization technologies that have emerged in the past decade. Containerization technologies allow the isolation of applications into separated instances with the binaries/libraries required. These isolate environments share an engine layer (commonly Docker) deployed in a host OS, avoiding have individual virtualized machines for each deployment. This technology has supposed a break with the old system deployment methods and has achieved great savings in terms of resources required because each container not require an entire operating system. Among many fields, Software testing exploits these containerization advantages, moreover with the last trends of moving the testing to the Cloud [1]. Despite the important savings achieved in testing due to the containerization, the execution of containerized E2E test suite is a challenging practice, due to the expensive and complex systems required, which arises the execution cost making unfeasible re-execute the suite as needed The contributions of this line of research are (1) A complete identification and characterization of the resources employed during E2E testing (2). A test case and resource organization to achieve an effective/efficient use of resources in E2E testing.

The remainder of this paper is organized as follows: the background and related work are described in Section 2, the research project overview is presented in Section 3 and finally, the conclusions and future work are exposed in Section 4.

2 Background and Related Work

E2E tests are those that test all application flow from start to end ensuring that all are working as expected. During the E2E test suite execution, it is common that the test cases require the entire system up for its execution, becoming costly in terms of both resources and time. Despite the inherent cost of re-deploy the resources of the system, developers usually face with another issue: the oversubscription. The oversubscription is caused by an overestimation or a bad optimization of the resources needed by the test case, instantiating more resources than needed. For example, one test oversubscribes resources if in order to check the data integrity into a database, deploys an unnecessarily complete web server that would not use.

If an E2E test suite is integrated into a continuous integration system without regarding the resource sharing/oversubscription the advantages achieved [2], would be overshadowed by the resources wasted in each repository change. The last tendencies of moving the testing to the Cloud [1], the on-demand resources and scalability, make disappear resource lacks, but increases the execution costs as often as the E2E suite is executed, because of the oversubscription.

The efficient and effective use of resources during testing has attracted the attention of both industry and academia. A number of works address this problem with prioritization, selection and minimization techniques [3]–[8], analyzing both fault detection rates and the total cost of executing the suite. Despite the positive results achieved by these kinds of techniques in large enterprises

as Google [8] or Microsoft [9], they raise particular challenges when dealing with E2E tests because the test cases deploy the whole system for their execution. Other authors have studied interdependences [10]–[15] between test cases in order to allow sharing resources between each other. Although the isolation achieved with the virtualization/containerization technologies, when the resources are costly, discover the dependencies is important to share those resources between the test cases.

3 Research Project Overview

This research project is intended to address efficient resource usage during E2E testing via characterization of the resources employed and a smart test case arrange with the resources required. The proposal, so-called RETORCH (Resource-aware orchestration framework for E2E testing) were preliminary published on [16] and is composed of three processes that are the basis of the orchestration technique: (1) the resource identification process characterizes the resources required, (2) the *grouping process* aims to reduce the costs, putting together the test cases with the same resource requirements to allow sharing and avoid unnecessary deployments, and (3) the *scheduling process* arrange the test cases focused on achieving savings in execution time

Resource Identification: The first process identifies the resources employed by each test case in order to detect which test cases require similar resources. These resources are classified in a number of different categories and are characterized by several *static attributes*, that describe how the resource is used and made available for the test. These include the *elasticity* that refers to the possibility of making available the resource on the fly, the *resource hierarchy* if the resource can be replaced by a mock during the test phase and a *Lifecycle* with different phases as set-up, execution or disposal. Resources are also characterized by a number of *access modes* such as *read*, *read-write*, *write-only* or *dynamic*. The access modes are related to how much safe and idempotence are the operations performed by the test cases over it, allowing sharing between the test cases that perform compatible use of those resources. Resources also have *dynamic attributes*, that change during the resource usage such as: *measurable* to refer that it has indicators to measure its performance, *traceable* that allows knowing in what phase of the lifecycle, or *availability* according to how and the number of times that may be instantiated, in order to specify how and the number of resources would be deployed.

For instance, in one IoT testing infrastructure, we have sensor simulators (containerized virtual devices), emulators (virtual devices but bound to physical interfaces) and physical devices. The *elasticity* of the three types could decrease as more physical is the device, being able to instantiated "on the fly" the simulators, and a fixed number of emulators-physical devices. The main objective of these emulators-simulators is to *replace* the real device (that is usually costly and non-flexible) performing the same type of *access mode* over it (read-only), and facilitate the performance *measuring* of the infrastructure (i.e. docker-stats).

Grouping Process: This process is aimed to optimize the usage of resources through an aggrupation of test cases according to the way they use such resources. These groups of test cases, called T-Groups, include all the scaffolding required for its execution and are focused to avoid the oversubscription and reducing the total cost of performing the test suite. For example, in the previous example of the IoT infrastructure, we can organize the test cases into T-Groups according to the different resource requirements (simulator, emulator or physical device) and the operations performed over it. This TGroup arrangement is focused on not deploy more resources than needed (i.e. a simple test that checks a sensor heartbeat, don't need an entire/expensive physical device

Scheduling Process: Finally, the T-Groups are optimized to achieve resource savings in terms of time. The T-Groups are divided, composed by a number of test cases and the scaffolding required (TJobs). The TJobs allow the parallelization and arrangement of the test case execution, as needed according to the execution constraints ensuring that the resources are deployed in an optimal way. Finally. in the previous IoT example, the TGroups are scheduled taking into account attributes as the flexibility or the access mode, in order to optimize the execution time of the entire test suite. Despite the test cases could be arranged in different feasible schedules (i.e. focused on reducing the time, the cost among others), the objective pursued is found the schedule that reduces the execution time at the same time it optimizes the resources employed.

The first preliminary results of this line of research were published on the QUATIC 19 conference [16] and has been invited to be extended for the Software Quality Journal. In [16], the approach was presented and exemplified with a real-world example of an educational application called Fullteaching [17]. With this application, we propose to make smart resource identification and orchestration, that avoid deploying all the functionality when it is not needed. Into the extension, we not only have achieved savings in terms of time (61% less than the non-orchestrated suite) but also a better resource usage in terms of physical memory (i.e. reducing the number of instances).

4 Conclusions and future work

The efficient usage of resources in E2E testing is a challenging and promising field. We have seen how with smart resource identification and organization of the test cases with the resources required, we achieve savings in terms of time and also reductions in the total cost of the test suite. As future work, we aim to extend the resource identification with a taxonomy and attributes provided, and automatize the organization, taking into account the dependencies between the different test cases and the resource properties, given by the identification process.

ACKNOWLEDGMENTS

This work was supported in part by the Spanish Ministry of Economy and Competitiveness under TestEAMoS (TIN2016-76956-C3-1-R) project and ERDF funds.

© C. Augusto 2020. This is the author's version of the work. It is posted here for your personal use. Not for redistribution. The definitive version was published in https://dl.acm.org/doi/10.1145/3377812.3382177, https://doi.org/10.1145/3377812.3382177

REFERENCES

[1] A. Bertolino *et al.*, "A Systematic Review on Cloud Testing."

[2] B. Vasilescu, Y. Yu, H. Wang, P. Devanbu, and V. Filkov, "Quality and productivity outcomes relating to continuous integration in GitHub," in *2015 10th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering, ESEC/FSE 2015 - Proceedings*, 2015, pp. 805–816.

[3] S. Yoo and M. Harman, "Regression testing minimization, selection and prioritization: A survey," *Software Testing Verification and Reliability*, vol. 22, no. 2. John Wiley and Sons Ltd., pp. 67–120, Mar-2012.

[4] R. Lachmann, M. Nieke, C. Seidl, I. Schaefer, and S. Schulze, "System-level test case prioritization using machine learning," in *Proceedings - 2016 15th IEEE International Conference on Machine Learning and Applications, ICMLA 2016*, 2017, pp. 361–368.

[5] O. Legunsen, F. Hariri, A. Shi, Y. Lu, L. Zhang, and D. Marinov, "An extensive study of static regression test selection in modern software evolution," in *Proceedings of the ACM SIGSOFT Symposium on the Foundations of Software Engineering*, 2016, vol. 13-18-Nove, pp. 583–594.

[6] G. Rothermel, M. J. Harrold, J. Von Ronne, and C. Hong, "Empirical studies of test-suite reduction," *Softw. Test. Verif. Reliab.*, vol. 12, no. 4, pp. 219–249, 2002.

[7] W. E. Wong, J. R. Morgan, S. London, and A. P. Mathur, "Effect of test set minimization on fault detection effectiveness," *Softw. - Pract. Exp.*, vol. 28, no. 4, pp. 347–369, 1998.

[8] A. Memon *et al.*, "Taming google-scale continuous testing," in *Proceedings - 2017 IEEE/ACM 39th International Conference on Software Engineering: Software Engineering in Practice Track, ICSE-SEIP 2017*, 2017, pp. 233–242.

[9] H. Esfahani *et al.*, "CloudBuild: Microsoft's distributed and caching build service," in *Proceedings - International Conference on Software Engineering*, Austin, Texas, 2016, pp. 11–20.

[10] A. Gambi, J. Bell, and A. Zeller, "Practical Test Dependency Detection," in *Proceedings - 2018 IEEE 11th International Conference on Software Testing, Verification and Validation, ICST 2018*, 2018, pp. 1–11.

[11] J. Bell, G. Kaiser, E. Melski, and M. Dattatreya, "Efficient dependency detection for safe Java test acceleration," in *2015 10th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering, ESEC/FSE 2015 - Proceedings*, Bergamo, Italy, 2015, pp. 770–781.

[12] A. Gyori, A. Shi, F. Hariri, and D. Marinov, "Reliable testing: Detecting state-polluting tests to prevent test dependency," in *2015 International Symposium on Software Testing and Analysis, ISSTA 2015 - Proceedings*, 2015, pp. 223–233.

[13] K. Muşlu, B. Soran, and J. Wuttke, "Finding bugs by isolating unit tests," in *SIGSOFT/FSE 2011 - Proceedings of the 19th ACM SIGSOFT Symposium on Foundations of Software Engineering*, 2011, pp. 496–499.

[14] S. Tahvili, L. Hatvani, M. Felderer, W. Afzal, and M. Bohlin, "Automated functional dependency detection between test cases using Doc2Vec and Clustering," in *Proceedings - 2019 IEEE International Conference on Artificial Intelligence Testing, AITest 2019*, 2019, pp. 19–26.

[15] T. Zimmermann and N. Nagappan, "Predicting defects using network analysis on dependency graphs," in *Proceedings - International Conference on Software Engineering*, 2008, pp. 531–540.

[16] C. Augusto, J. Morán, A. Bertolino, C. de la Riva, and J. Tuya, "RETORCH: Resource-aware End-to-end Test Orchestration," in *12th International Conference on the Quality of Information and Communications Technology (QUATIC 2019)*, 2019, p. 14.

[17] P. F. Pérez, "Fullteaching: A web application to make teaching online easy." Universidad Rey Juan Carlos, 2017.