Toward an efficient End-to-End test suite execution

Cristian Augusto†
Computer Science Department
University of Oviedo
Gijón, Asturias (Spain)
augustocristian@uniovi.es

***Abstract*—End-to-end (E2E) testing is costly because of the complex and expensive resources that are required during the test execution coupled with the long execution times required. This becomes even more challenging when E2E test suites are integrated into a continuous integration (CI/CD) environment, where they are executed with each repository change. Migrating E2E test suite execution to the Cloud is an acknowledged trend to achieve a better cost. However, this also introduces new challenges in addition to those faced on-premises, such as selecting the most suitable services from the wide range offered by Cloud Providers, which is even more difficult considering how the test resources use the Cloud infrastructure. This thesis aims to achieve an efficient execution of the E2E test suites, reducing the number of unnecessary test resource redeployments, and the execution time, and improving the efficiency of selecting the Cloud infrastructure that best aligns with the testing objectives. We present an orchestration approach that aims to enable resource sharing and avoid unnecessary redeployments. This process involves phases like the characterization of the test resources required by the test cases and a grouping of the test cases with compatible resource usage and its scheduling in sequential-parallel to reduce redeployments-time. The orchestration approach has evolved to execute this orchestrated test suite in the Cloud, introducing a model of the E2E test suite execution in the Cloud, which allows us to represent both the test and the Cloud configuration. Using the model, we estimate and compare different Cloud infrastructures in terms of overall cost (billed by the Cloud Provider), but also the cost invested in testing and unused infrastructure (overprovisioning) to select the infrastructure best aligned with the testing objectives.**

Keywords— Software Reliability, Software Testing, Cloud Computing, End-to-End Testing, Resource Optimization

I. MOTIVATION AND RELATED WORK

Continuous integration (CI) practices have shortened development cycles from months or years to weeks or days, impacting critical stages such as software testing. Modern software developments have test suites composed of thousands of test cases that are executed frequently [1], [2]. However, handling shorter cycles may result in that test suite not being executed as often as required. which could cause potential defects to be missed, with an impact of 20-40% extra time and cost [3].

E2E testing validates the entire interaction between all system components, from the user interaction with the application to low-level layers like persistence. The execution of E2E test suites is expensive due to long execution times, expensive test resources (e.g., complex databases, web servers, or web browsers), and due to the fact that it requires the entire system during its execution.

Test suite optimization techniques arise as a solution for reducing both time and cost in large and complex test suites and enabling their execution as needed. The traditional techniques, such as test prioritization, selection, and minimization [4] have shown good performance in detecting defects in both industry and academia [5]–[7] while optimizing execution cost/time.

However, when it comes to End-to-End (E2E) testing, these traditional test optimization techniques [4] may not be as effective as expected because a subset or order of the test suite might still require the use of the same expensive system for its execution. Consequently, further research is required in this area, and alternative solutions such as test dependency detection to organize test cases and enable resource sharing [8], [9] or the execution of tests on a Cloud infrastructure [10] emerge to address these limitations.

Moving testing to the Cloud means that the infrastructure for test execution is contracted over the Internet as-a-service and has been widely acknowledged as a way to reduce the cost of testing whilst leveraging the unlimited and scalable infrastructure delivered on-demand [11]. The Cloud is suitable for those testing levels that are not continuously executed but require large amounts of infrastructure, such as concurrency, load [12], or E2E testing. However, the efficient execution of E2E test suites in the Cloud remains challenging, as the test resources required to run E2E test suites do not always use the full contracted infrastructure and there is also a broad range of infrastructures to deploy the same test resource.

The selection of Cloud infrastructure (also referred to as highly configurable systems) has been widely discussed in the literature where several tools [13]–[15] have been proposed to choose the lower-cost infrastructure under certain requirements (e.g., memory size, number of processors, or storage). These tools are useful with homogeneous loads but, the execution of E2E testing in which load peaks are addressed differently and its usage patterns differ from an application deployed in production reduces its effectiveness.

II. RESEARCH QUESTIONS

During the Ph.D., we intend to answer the following research questions:

- **RQ1:** Can a smart characterization, aggrupation, and scheduling of the test cases achieve savings in resource usage?
- **RQ2:** How does the cost/efficiency of executing an E2E test suite in the Cloud vary depending on the Cloud Infrastructure and the test suite scheduling selected?

III. PLANNED CONTRIBUTIONS

Upon completion of the Ph.D., we intend to make the following contributions:

- **Characterization of resources** required by E2E test cases with a set of attributes that describe how they can be used by the test cases and their access modes. The goal of the characterization is to provide valuable information about the nature of the test resources as well as the operations performed by the test cases on them.
- An **E2E Test Execution Orchestration approach** that identifies the test resources required by the test cases, groups them according to their compatible usage, and schedules them sequentially or in parallel to reduce time/unnecessary redeployments (RQ 1).
- **Cost Model of E2E test suite execution in the Cloud** that considers the overall cost (contracted) as well as the cost invested in executing the test suite (testing) and unused infrastructure (overprovisioned). The model goal is to compare several Cloud infrastructures and select those that best align with the testing objectives (RQ 2).

IV. APPROACH

The approach is divided into two distinct parts: orchestration and cost model. The orchestration part is responsible for arranging the test cases in the most efficient order, aiming to reduce both cost and time. The cost model focuses on selecting the most suitable Cloud infrastructure that best aligns with the testing objectives. These objectives may include cost reduction, time optimization, or improving infrastructure utilization. The following subsections will introduce and describe these two components in detail:

A. RETORCH

The characterization of the test resources required by E2E test cases is carried out through a set of attributes and access modes to provide additional information about the test resources and their usage by the test cases. Each test resource has one or more attributes that provide information such as the number of available test resources, concurrent access support, or hierarchical relationships between test resources. The access modes indicate how the test cases use the test resources, such as *read-only, read-write, write-only, or no-access*.

The test resource characterization is the basis of the proposed orchestration approach named RETORCH, which stands for "*Resource-aware End-to-end Test ORCHestration framework*" and consists of a four-phase process: resource identification, grouping, scheduling, and deployment. RETORCH aims to optimize the execution of the E2E test suite by reducing execution time and test resource redeployments.

Spanish Ministry of Science and Innovation under TestBUS (PID2019-105455GB-C32) and EQUAVEL (PID2022-137646OB-C32)

The first phase of the orchestration approach (Figure 1, resource identification), receives the E2E test cases as input and characterizes and annotates the test resources used in each test case. In the second and third phases (Figure 1, grouping and scheduling), the test cases with compatible test resource usage (e.g., their execution on the same test resource does not affect the execution of other members of the group) are grouped in the so-called TJobs. The TJobs contain the test cases with compatible resource usage and the test resources required for their execution and are scheduled in sequential/parallel in the so-called Execution Plan. The fourth phase (Figure 1, deployment) gets as output the required script and code to execute the Execution Plan through a continuous integration system.

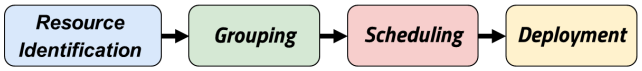

Figure 1 RETORCH Orchestration Approach

B. Cost Model

The Execution Plan provided by RETORCH can be deployed in both on-premises and Cloud infrastructures. Deploying it in a Cloud infrastructure allows to achieve better cost, but it also introduces new challenges on top of those already faced on-premises. These challenges include selecting the proper test configuration and the best allocation of test resources; a process that becomes even more difficult with the broad range of infrastructure types offered in the different providers to deploy the same test resource.

In response to these challenges, we introduce a Cloud-based E2E test execution model called RETORCH* (to distinguish it from the RETORCH proposal). RETORCH* aims to enable the comparison of Cloud infrastructures in terms of the cost of executing a certain test execution plan. The RETORCH* model is composed of two different submodels: the Test Configuration and the Cloud Configuration. The Test Configuration represents the different test entities (e.g., test resources, test cases) and their attributes (e.g., how the test cases are arranged or what type of operations are performed over the test resources). The Cloud Configuration represents all feasible combinations of Cloud Objects, e.g., the different virtual computing environments contracted in the Cloud, under certain Billing Options that can be used as the execution platform of the test suite.

Based on the information provided by the Test and Cloud Configuration, we estimate and compare the cost of executing the test suite in the different Cloud infrastructures, not only in terms of the cost that is billed by the Cloud Providers (**overall**), but also the cost invested in testing (**testing cost**) and infrastructure not used (**overprovision**).

The testing cost, in turn, is composed of three different costs: the **set-up** and **tear-down** costs of both Cloud Object Instances (COI) and TJobs and the **test execution** cost. These costs are calculated according to the capacities invested in the different phases, some of them do not execute the test cases themselves but impact the billing as they are required before the execution and differ in duration, depending on the selected COI. For instance, setting up all the infrastructure for a virtual machine takes more time compared to a container. Moreover, it takes even longer than a service that provides the test resource with no

set-up. To measure them, we have presented two lifecycles that allow a fine analysis of how the infrastructure is used:

- **COI lifecycle** consists of three stages: the *set-up* (gray in Figure 2) where the COI is provisioned, and the necessary libraries and dependencies are installed, the *TJob execution* (red border in Figure 2) that executes the TJobs in the order specified by the Execution Plan. Finally, the *tear-down* (violet in Figure 2), involves releasing the COI and returning it to the Cloud Provider.
- **TJob's lifecycle** consists of three distinct stages: the *set-up* (yellow in Figure 2) instantiates the required test resources and performs any required actions to prepare them for testing. Next, the *test execution* (orange in Figure 2) executes the test cases against the previously instantiated test resources and finally, the *tear-down* (green in Figure 2) performs the cleaning and disposing actions.

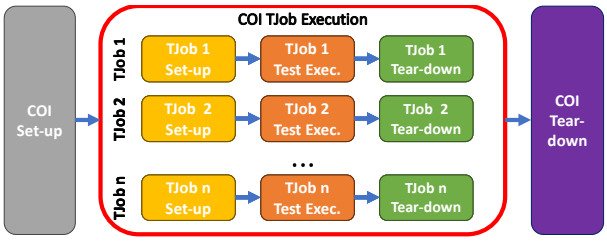

Figure 2 TJob and COI Lifecycles

The estimation of the E2E test execution cost is done according to how the test resources required by the E2E test suite use the Cloud Object capacities (the different specifications that are contracted under a billing option, such as the amount of memory, the number of processors, or storage) during the lifecycle's phases. Each of these capacities has a **utilization** carried out by the test resources, which means the percentage of used contracted capacity. The rest of the (unused) capacity is allocated to **overprovisioning**, which results in an extra budget cost that is wasted or can be used for other test suites/executions.

Both, the **utilization** and **overprovisioning** are displayed graphically in the **usage profile**, which represents the utilization of the different capacities of the COI during the time that is provisioned. Figure 2 depicts an example of a profile in which two TJobs are executed in parallel. The profile shows the capacity usage (GB of memory in the x-axis) during the time that the Cloud Object is provisioned (60 seconds, in the y-axis)

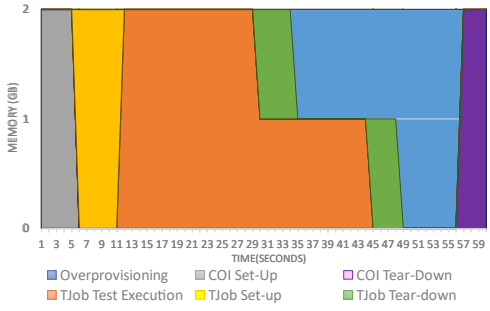

Figure 3 Usage Profile example

Using the information provided by the profile, we can estimate all costs. For instance, if the price of each GB of RAM/s is $2, the costs are calculated as follows:

- The overall cost is calculated according to the amount of capacity contracted 60s x 2GB multiplied by the invoiced price of $2/GBs, resulting in a total of $240.
- The testing cost is the sum of all colored areas (excluding the blue area) which results in 90GB*s multiplied by its invoiced price ($2/GB*s), totaling $180.
- The overprovisioning cost is calculated as the difference between both areas: overall (120G*s) and testing (90GB*s) multiplied by the invoiced price ($2/GB), resulting in a total of $60.

With the different costs, the tester compares the different Cloud infrastructures and selects which is best aligned with the organization's testing objectives. For example, larger enterprises may have already contracted infrastructures that can be reused, smaller ones may be interested in deploying as cheaply as possible, while some enterprises may opt to contract extra capacities for reuse in other tasks.

V. Progress and Preliminary Results

A. *Characterization of resources required by E2E test cases.*

We have developed a characterization of the test resources used in the E2E tests which is composed of five distinct access modes, seven dynamic attributes that change during execution as a result of the resource usage, and five static attributes which provide additional information regarding how each test resource can be used. The resource characterization is continuously enhanced with new attributes, currently following the trend to migrate the testing to the Cloud. It has been extended with those attributes required to represent all information about how the test resource uses the Cloud Infrastructure.

B. *RETORCH Orchestration approach*

The first RETORCH experimental prototype was a result of my Master's Thesis [16]. This RETORCH prototype analyzes the test resources that are manually annotated in the test cases, to group and schedule them maximizing the use of the available test resources, giving as output the scripts and code that automates the execution. RETORCH was validated in a real-world application called *Fullteaching* [17], achieving 65% reductions in terms of execution time and 35% in terms of test resources. Both the characterization and the orchestration approach were first introduced at the *QUATIC19* [18] conference, extended to *Software Quality Journal* [19], and presented in the *ICSE20 ACM Research Competition* [20].

The prototype is intended to be enhanced through the automation of the resource identification process by a smart identification of the test resources required by E2E test cases using the test code analysis.

C. *RETORCH* Cost Model of E2E test suite execution in the Cloud*

RETORCH* is being validated into a real-world example (*FullTeaching*), exploring the three major decisions that impact

the cost and efficiency of carrying out a test suite in the Cloud: (1) selecting the Cloud Objects; (2) choosing the Billing Options; and (3) determining the Execution Plan.

RETORCH* evaluation is being performed through three case studies: The first study explores different types of Cloud Objects, representing the most common offerings in the industry, for a fixed Execution Plan and Billing Options. The second study examined different Billing Options for the selected Cloud Objects mentioned earlier. Lastly, in the third case study, we explore different Execution Plans with different TJob arrangements.

The preliminary results demonstrate how RETORCH* provides cost estimations that reveal differences between infrastructure alternatives. Interestingly, the decisions taken relying only on the overall cost might contradict each other if we consider the rest of the costs. For instance, Cloud Objects that appear more expensive in terms of overall cost achieve the best testing cost and offer extra capacities for use with other executions/test suites, or using the cost estimation, the tester can devise the cost of executing the test suite in already contracted infrastructure which is shared for other purposes.

RETORCH * is intended to be the core of an advisory engine that receives the data generated in the CI Environment (e.g., execution times, infrastructure usage time, or overprovisioning) and suggests infrastructure or test configuration changes to improve its execution in the Cloud.

Tentative date of Ph.D. Thesis defense.

The author intends to defend his Ph.D. thesis in the middle of the year 2024.

VI. ACKNOWLEDGMENTS

The author would like to express sincere gratitude to his Ph.D. Advisor, Dr. Claudio de la Riva, for his dedicated effort and guidance throughout the thesis direction. Additionally, special thanks go to Dr. Antonia Bertolino (ISTI-CNR, Pisa Italia) for her contributions to the line of research. This work was supported in part by the *Spanish Ministry of Science and Innovation* under *TestBUS* (PID2019-105455GB-C32) and *EQUAVEL* (PID2022-137646OB-C32)

VII. REFERENCES

[1] 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, doi: 10.1145/2889160.2889222.

[2] 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*, May 2017, pp. 233–242, doi: 10.1109/ICSE-SEIP.2017.16.

[3] F. Shull *et al.*, "What we have learned about fighting defects," in *Proceedings - International Software Metrics Symposium*, 2002, vol. 2002-Janua, pp. 249–258, doi: 10.1109/METRIC.2002.1011343.

[4] 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, doi: 10.1002/stv.430.

[5] E. Engström, M. Skoglund, and P. Runeson, "Empirical evaluations of regression test selection techniques: A systematic review," in *ESEM'08: Proceedings of the 2008 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement*, 2008, pp. 22–31, doi: 10.1145/1414004.1414011.

[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, doi: 10.1002/stvr.256.

[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, doi: 10.1002/(SICI)1097-024X(19980410)28:4<347::AID-SPE145>3.0.CO;2-L.

[8] 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, doi: 10.1145/2771783.2771793.

[9] A. Gambi, J. Bell, and A. Zeller, "Practical Test Dependency Detection," *Proc. - 2018 IEEE 11th Int. Conf. Softw. Testing, Verif. Validation, ICST 2018*, pp. 1–11, 2018, doi: 10.1109/ICST.2018.00011.

[10] A. Gambi, A. Gorla, and A. Zeller, "O!Snap: Cost-Efficient Testing in the Cloud," *Proc. - 10th IEEE Int. Conf. Softw. Testing, Verif. Validation, ICST 2017*, pp. 454–459, 2017, doi: 10.1109/ICST.2017.51.

[11] A. Bertolino *et al.*, "A systematic review on cloud testing," *ACM Comput. Surv.*, vol. 52, no. 5, 2019, doi: 10.1145/3331447.

[12] K. Inçki, I. Ari, and H. Sözer, "A survey of software testing in the cloud," *Proc. 2012 IEEE 6th Int. Conf. Softw. Secur. Reliab. Companion, SERE-C 2012*, pp. 18–23, 2012, doi: 10.1109/SERE-C.2012.32.

[13] J. García-Galán, J. M. García, P. Trinidad, and P. Fernández, "Modelling and analysing highly-configurable services," in *ACM International Conference Proceeding Series*, 2017, vol. 1, pp. 114–122, doi: 10.1145/3106195.3106211.

[14] J. García-Galán, O. Rana, P. Trinidad, and A. Ruiz-Cortés, "Migrating to the Cloud: A software product line based analysis," 2013. doi: 10.5220/0004357104160426.

[15] C. Plewnia, "An integrated approach for cloud computing service selection and cost estimation," Dec. 2021, doi: 10.1145/3492323.3503505.

[16] C. Augusto, "Optimización de Recursos en Pruebas de Sistema," Universidad de Oviedo, Gijón (Asturias), Spain, 2020.

[17] URJC and P. F. Pérez, "Repositorio FullTeaching - ElasTest." 2019, [Online]. Available: https://github.com/elastest/full-teaching.

[18] C. Augusto, J. Morán, A. Bertolino, C. de la Riva, and J. Tuya, "RETORCH: Resource-Aware End-to-End Test Orchestration," in *Communications in Computer and Information Science*, Sep. 2019, vol. 1010, pp. 297–310, doi: 10.1007/978-3-030-29238-6_22.

[19] C. Augusto, J. Morán, A. Bertolino, C. de la Riva, and J. Tuya, "RETORCH: an approach for resource-aware orchestration of end-to-end test cases," *Softw. Qual. J.*, no. SQJO-D-19-00197R1, 2020, doi: 10.1007/s11219-020-09505-2.

[20] C. Augusto, "Efficient test execution in End to End testing : Resource optimization in End to End testing through a smart resource characterization and orchestration," in *Proceedings - 2020 ACM/IEEE 42nd International Conference on Software Engineering: Companion, ICSE-Companion 2020*, 2020, pp. 152–154, doi: 10.1145/3377812.3382177.