Contents lists available at ScienceDirect

The Journal of Systems & Software

journal homepage: www.elsevier.com/locate/jss

In Practice

RETORCH*: A Cost and Resource aware Model for E2E Testing in the Cloud

Cristian Augusto[a,*], Jesús Morán[a], Antonia Bertolino[b], Claudio de la Riva[a], Javier Tuya[a]

[a] *Computer Science Department, University of Oviedo, Gijón, Spain*
[b] *ISTI-CNR, Consiglio Nazionale delle Ricerche, Pisa, Italy*

ARTICLE INFO

Editor: Prof J. C. Duenas

Keywords:
Software engineering
Test optimization
Software testing
Cloud computing
End-to-end testing
E2E testing

ABSTRACT

Moving testing to the Cloud overcomes time/resource constraints by leveraging an unlimited and elastic infrastructure, especially for testing levels like End-to-End (E2E) that require a high number of resources and/or execution time. However, it introduces new challenges to those already faced on-premises, like selecting the most suitable Cloud infrastructure and billing scheme. We propose the RETORCH* test execution model that estimates and compares the monetary cost of executing an E2E test suite with different Cloud alternatives, billing schemes, and test configurations. RETORCH* goes beyond the mere cost billed, and selects the solution that best aligns with the test team strategy using the data of on-premises prior executions and the tester's experience. This cost is broken down into the cost incurred to execute the test suite (testing cost) and possible unused infrastructure (overprovisioning cost). Based on these distinct costs, the test team can compare different Cloud and test configurations. RETORCH* has been evaluated using a real-world application's E2E test suite. We analyze how the different decisions taken when the suite is migrated to the Cloud impact the cost, highlighting how RETORCH* can help the tester during Cloud and test configuration to make a more informed decision.

1. Introduction

Moving test suite execution to the Cloud is an acknowledged trend (Bertolino et al., 2019) to overcome testing resources constraints by leveraging the potentially unlimited and scalable infrastructure. This trend, even more, applies to resource intensive testing levels like End-to-End (E2E), which validate all interactions between system components, from end-user interactions down to low-level components. This validation requires significant Resources[1] such as web servers, browsers, or large databases, which in turn results in further use of resources due to the high execution time, the cost of replicating Resources, or the set-up of the system, among others.

Cloud testing removes the resource limit barriers for E2E test execution, but puts new challenges on top of the ones already faced on-premises, also because the E2E testing infrastructure usage patterns differ from a standard application in production. Typically, the Cloud infrastructure is shared among multiple test teams and projects, and E2E test suites undergo multiple executions, resulting in a non-uniform utilization of the Cloud infrastructure with usage peaks. Furthermore, the Cloud providers offer a broad range of different infrastructure alternatives with heterogeneous billing schemes that are not always aligned with the E2E test execution requirements, leaving the testers in "*the agony of choice*" (Zhang et al., 2012) among all these alternatives.

In prior work (Augusto et al., 2020) we addressed the efficient execution of E2E test suites through an E2E test orchestration approach called RETORCH, which optimized the test configuration and execution ordering based on a proper characterization of the Resources required by the test cases. This optimized configuration involved sharing Resources among compatible test cases, either by sharing the entire System Under Test (SUT) or specific parts of it. Alternatively, those test cases that might otherwise produce collateral effects (Gyori et al., 2015) on the rest of the test suite were executed in isolation.

In this article, we extend our previous work (Augusto et al., 2020) by considering a new factor: the cost of executing the E2E test suite in the Cloud. This factor is used to optimize the E2E test execution, through the estimation of the monetary costs of executing the E2E test suite with different test configurations (e.g., test case arrangements), and Cloud alternatives (e.g., virtual machines, containers) under different contractual schemes. For clarity, we refer to the new extended model as RETORCH*, whereas the original term RETORCH is used when expressly

* Corresponding author.
E-mail addresses: augustocristian@uniovi.es (C. Augusto), moranjesus@uniovi.es (J. Morán), antonia.bertolino@isti.cnr.it (A. Bertolino), claudio@uniovi.es (C. de la Riva), tuya@uniovi.es (J. Tuya).

[1] Henceforth, we will use the term "Resources" (capitalized) when referring to the ones required by the E2E test suite.

https://doi.org/10.1016/j.jss.2024.112237
Received 13 August 2024; Accepted 2 October 2024
Available online 3 October 2024

referring to the previous version (Augusto et al., 2020).

While there exists a wide literature on managing and scheduling Cloud resources (García-Galán et al., 2017; García Galán, 2015), to the best of our knowledge this is the first work that focuses specifically on how to estimate the costs of executing the E2E test suites in a Cloud Infrastructure. The construction and validation of the RETORCH* model relies on the one side on our long-term expertise in testing approaches and strategies, and on the other side on an extensive analysis and abstraction of the different offerings from the most popular Could providers. RETORCH* combines the two perspectives into one model specifically tailored for E2E testing and supports a test team with an easy visual illustration (called the *Usage Profile*) of the possible costs for different alternative solutions. It is worth noting that RETORCH* does not incur any billing cost to make the estimation, since it does not require the actual deployment of the alternatives in the Cloud: it uses past data (from on-premises or other Cloud executions) and an easy tuning by the tester. After comparing the different alternatives, RETORCH* not solely provides the overall cost of Cloud (as the state-of-the-art tools discussed in Section 5.3), but also the cost of executing the test suite and the amount of unused infrastructure (overprovisioning). These different costs for each Cloud alternative, billing scheme, and test configuration are provided to the tester who can thus select the one that is more aligned with the test team's objectives. More specifically, this article provides the following contributions:

(1) An End-to-End Test Execution Model in the Cloud, which enables the representation of the Cloud Configuration with different options of Cloud infrastructure and billing schemes and the Test Configuration with the different options related to the test execution.
(2) The cost estimation of an E2E test suite execution in the Cloud, considering not only the Overall cost (contracted) but also the Test execution, Set-up, Tear-down, and Overprovisioning costs, which enables the comparison of different Cloud Infrastructures and Test configurations.
(3) An evaluation of the model comparing different Cloud infrastructures and test configurations, allowing the tester to analyze the impact of selecting different Cloud alternatives, billing schemes, test configurations, and Cloud Infrastructure sharing.

The remainder of this paper is organized as follows: In Section 2, we introduce RETORCH* and provide the necessary background information about the previous RETORCH model. Section 3 further presents the cost model and how it works. In Section 4, we explain the methodology used in evaluating RETORCH*, including the research questions, the results, and the threats to validity. Sections 5 presents the related work. Finally, in Section 6, we present the conclusions drawn from our study and outline future directions for research.

2. RETORCH* model

The RETORCH* E2E Test Execution Model in the Cloud is composed by the Test Orchestration Submodel (TOSM) and the Cloud Configuration Submodel (CCSM). The former represents the E2E test cases, the test Resources they require, and how they are arranged during the E2E test suite execution, extending in several aspects our RETORCH original model (Augusto et al., 2020) (summarized below in Section 2.1).

The complete RETORCH* model is depicted in Fig. 1, with TOSM in gray color (dotted line) and CCSM in blue color (continuous line).

2.1. Background

To make the paper self-contained, we briefly recall the RETORCH E2E test orchestration technique (Augusto et al., 2020), which allows for optimizing the execution of E2E test suites based on the proper characterization of the test Resources required for their execution.

RETORCH relies on three important concepts, namely:

- **Resources** are physical, logical, or computational entities that are required during the execution of the E2E test cases. The Resources are characterized by a series of static and dynamic **attributes**, which provide additional information about each Resource and how it can be used.
- **TJobs** are a set of test cases with compatible Resource usage inside a container[2] that also provides the environment and containerized Resources composing the **S**ystem **U**nder **T**est (SUT)
- **Execution Plan** is the sequence of TJobs that is given as output by RETORCH. The different TJobs are organized in sequential or parallel ordering based on the Resource usage.

RETORCH gets as input the test cases, groups them according to their usage of Resources, and provides in output an Execution Plan. Fig. 2 depicts the steps of such a process: the test cases are characterized by their different attributes and access modes in the **Resource Identification** phase. Then these test cases are grouped according to their compatible Resource usage and scheduled based on their group assignments, forming the **TJobs** in the **Grouping and Scheduling** phase (e.g., two test cases that use the same Resource without making modifications can be executed together, while another test case that changes the same Resource and might affect the previous ones should be executed in isolation). Finally, in the **Deployment** phase the TJobs are deployed into a Continuous Integration Environment like Travis, Jenkins, GitLab CI/CD, or Azure Pipelines.

2.2. Test Orchestration Submodel

As mentioned above, the Test Orchestration Submodel extends the original RETORCH model (Augusto et al., 2020) aiming to identify the additional entities/attributes required to move the E2E test suite execution to the Cloud.

2.2.1. TJobs

The TJobs are extended with a lifecycle that comprises three distinct phases (Set-up, Test Execution, and Tear-down), as illustrated in Fig. 3.

During the **TJob Set-up** phase, the SUT is prepared for execution by deploying the Resources within the infrastructure. Once everything is ready, the **Test Execution** phase begins, during which the test cases are executed against the SUT. Finally, in the **TJob Tear-down** phase, all necessary cleaning actions to ensure that the environment is left in a proper state are performed. These stages were not distinguished in the original RETORCH approach; here we need them for a more precise assessment of Cloud utilization costs.

2.2.2. Resources

In RETORCH*, for deploying a Resource in a specific infrastructure some requirements have to be fulfilled to ensure the Resource's proper operation. By **Minimal Capacities** we denote the infrastructure specifications required to deploy a single Resource. The Minimal Capacities have a *category* (such as memory, processors, or storage) and a *size* (e.g., 1vCPU, 1 GB of memory, or 1 TB of storage). The Minimal Capacities in combination with The TJob lifecycles not only enable us to size the contracted Cloud Infrastructure, but also devise how this infrastructure is used.

We extend the concept of Resource with the definition of the **Resource Instance** that serves as connection between the Test Configuration and the Cloud Configuration model. Resource Instances are

[2] A lightweight virtualized environment with only the essential binaries and libraries required for the application's execution. See more (Koskinen et al., 2019)

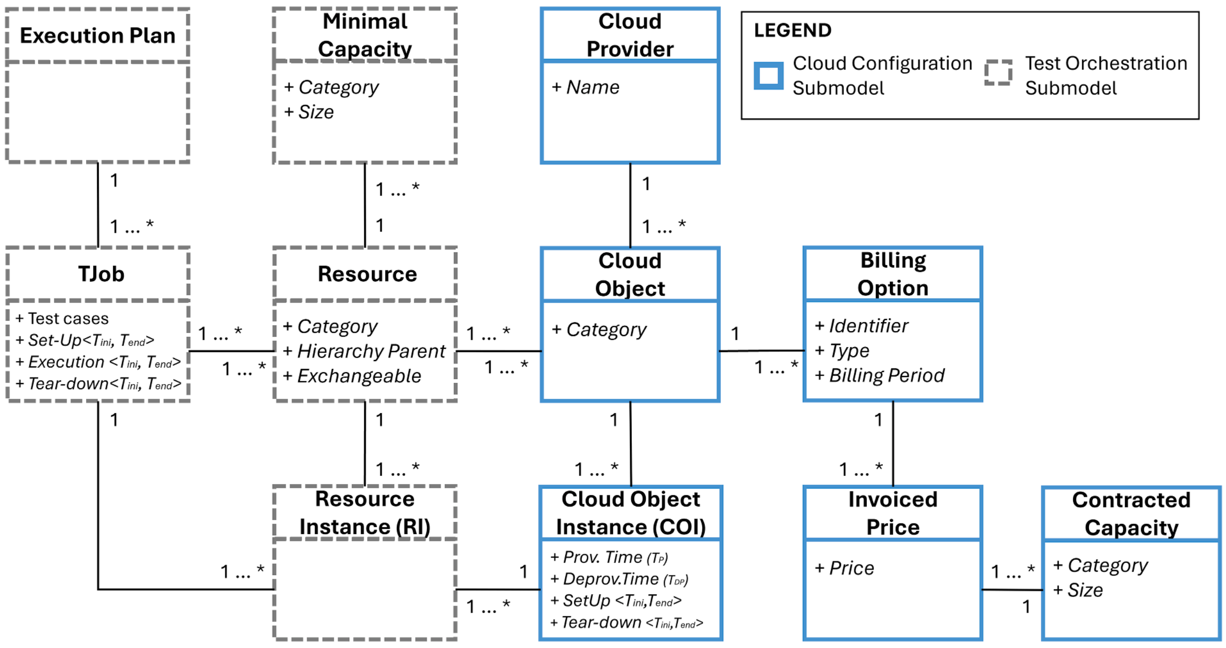


Fig. 1. RETORCH* Model of the E2E Test Execution in the Cloud.

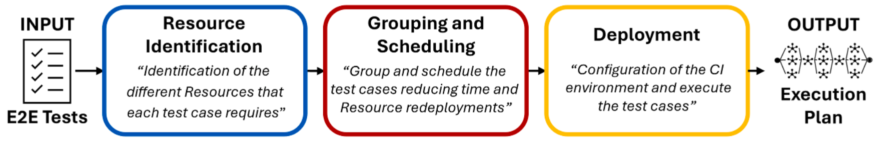

INPUT
E2E Tests
Resource Identification
"Identification of the different Resources that each test case requires"
Grouping and Scheduling
"Group and schedule the test cases reducing time and Resource redeployments"
Deployment
"Configuration of the CI environment and execute the test cases"
OUTPUT
Execution Plan

Fig. 2. RETORCH orchestration technique.

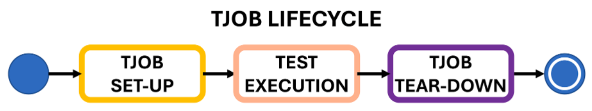


Fig. 3. TJob lifecycle phases.

Resources hosted into a specific infrastructure (either on-premises or in the Cloud) for use by a single TJob. A Resource can be instantiated multiple times, creating a single Resource instance for each TJob that requires it. For example, if several TJobs require the same database with users, the Resource is instantiated in several Resource Instances, one for each TJob.

2.3. *Cloud Configuration Submodel*

The Cloud Configuration Submodel (CCSM) is depicted in blue color in Fig. 1 and represents the Cloud infrastructure over which the TJobs are going to be deployed according to the different Resource Instances that are required during the Execution Plan. The CCSM represents the infrastructure characteristics, Cloud Provider, and billing information; it enables cost estimation with a series of new entities detailed in the following subsections.

2.3.1. *Cloud Objects*

Cloud Objects are virtual computing environments over the Internet that are contracted with a Cloud Provider and used as execution platforms for the TJobs. To construct RETORCH*, we comprehensively examined the different types of *Infrastructure, Platform, and Software as Service* offerings provided by prominent providers such as *Alibaba Cloud* (Aliyun 2023), *Amazon Web Services (AWS)* (Amazon 2023), *Google Cloud* (Google 2023), *Digital Ocean* (Digital Ocean 2023) and *IBM Cloud* (IBM 2023). Through our analysis, we identified three primary ***categories*** among these offerings:

1. **Virtual Machines** are digital versions of a physical computer on which an entire operating system and binaries are installed. Virtual Machines (also referred as instances (Aliyun, 2023; Amazon, 2023a, Google, 2023; Digital Ocean, 2023; IBM, 2023)) are the most common *Infrastructure as a Service (IaaS)*. The Virtual Machines are usually classified based on their size and/or intended purpose, examples are *"bx2–4x16"* and *"cx2d-8x16"* in *IBM Cloud, "t2-nano"* and *"t2-small"* in *AWS*, or *"ecs.g8a.large"* and *"ecs.g7se.xlarge"* in *Alibaba Cloud*. Virtual Machines require the highest degree of management in the Cloud, for maintaining the whole operating system along with the platforms required to deploy Resources such as *Docker*, web servers, or applications.
2. **Containers** are lightweight packages of application code and of the dependencies required to deploy a Resource. Considered also as a distinct Cloud service known as *Container as a Service (CaaS)* (Piraghaj et al., 2017), they require a lower degree of management by requiring users only to handle the dependencies and installed libraries.
3. **Services** are self-maintained Resources that are fully managed by the Cloud Providers and delivered on demand for their direct usage in the E2E test suite execution. Services are part of the *Software as a Service (SaaS)* option and entail the minimal degree of management.

2.3.2. *Billing Option*

Billing options represent the different ways in which the Cloud offerings are charged by one Cloud Provider in terms of time and money. A Billing Option consists of an **Invoiced Price** for each Contracted

Capacity and a **Billing Period,** which is the minimal time slot for which a Cloud Object can be bought. In RETORCH* depending on the *billing period,* we differentiate between two options:

1. **Pay-as-you-go (on-demand):** users pay only for the time a Cloud Object is provisioned and used, which is characterized by short *billing periods* (e.g., milliseconds or seconds).
2. **Pre-Invoiced (subscription)**: this option requires a reservation and advance payment for a longer *billing period* (hourly, daily, monthly, or yearly). This Billing Option usually includes discounts for reservations but does not allow users to release the Cloud Object in advance, i.e., they also pay for unused time.

2.3.3. Cloud Object Instances

The Cloud Object Instances (COI) are Cloud Objects contracted under a specific Billing Option from a Cloud Provider and used by the Resource Instances of one to multiple TJobs during the execution of the Execution Plan. The Cloud Object Instances have a lifecycle of three phases (depicted in Fig. 4), in which we distinguish the Contracted Capacities and time used for the Set-up, the TJob Execution and the Tear-down of the Cloud Object Instance.

More in detail the lifecycle phases include:

- **Provisioning and de-provisioning:** they are the start and end, respectively, of the lifecycle, in which the Cloud Object Instance is reserved under a Billing Option and hence released. Both phases are defined in each Cloud Object Instance with the *Provisioning time* and *Deprovisioning time*.
- **COI Set-Up:** during this phase, the entire Cloud Object Instance is blocked for installing the necessary tools and performing the different configurations. The duration of the set-up depends on the *type* of Cloud Object used, e.g., one virtual machine requires several configuration steps, such as installing a monitoring platform or a containerization engine, while a service provides Resources on-the-fly with a minimal set-up time.
- **TJob Execution**: once the Cloud Object Instance is ready, TJobs using it can start deploying the different Resource Instances either sequentially or in parallel. Each TJob during this phase performs its self-set-up, executes the test cases until the last test has finished and perform its own necessary tear-down actions. The start of this phase is defined by the Set-up of the earlier TJob and the Tear-down of the last TJob.
- **COI Tear-down**: during this phase, the Cloud Object Instances are shut down, and several actions such as saving the different test outputs or logs are performed before the de-provisioning of the Cloud Object Instance.

2.3.4. Contracted Capacities

The **Contracted Capacities** are the different Cloud Object specifications that are contracted under a Billing Option and used during the execution of the Execution Plan.

Each Contracted Capacity belongs to a **category**; in the CCSM we consider memory, storage, processors, graphical units, and slots (logical partitions used to deploy fully managed Resource Instances). The Contracted Capacities also have a **size** that represents the amount available. Depending on the Cloud Provider (Azure, AWS, IBM among others) and Object Contracted, the Contracted Capacity can be offered in a fixed range (e.g., a *t2.nano* is a virtual machine with 1 vCPU and 0.5 GB of memory) or can be fine-tuned (e.g., a custom virtual machine with 200 MB of memory and 0.5 vCPU). Below are the common Contracted Capacities available with their sizes and categories in most of the Cloud Providers and Objects:

1. **Virtual machines** are usually contracted with four different Contracted Capacities categories: the (1) *memory* and (2) *storage* categories, sized in Mb/GB, the (3) *processors* and (4) *graphical units* sized in a number of cores or processors.
2. **Containers** are contracted with three different Contracted Capacities *categories*: the *memory size* in Mb/GB, the *processors* and *graphical units* sized in a number of cores or processors.
3. **Services** are heterogeneous and each is offered differently. In RETORCH we focus on one single Contracted Capacity: number of Resource Instances that are provided. We refer to them as **slots**, whereby one *slot* is a logical partition of the service on which the provider deploys on-the-fly a fully managed Resource Instance ready for usage. For example, Aerokube (the enterprise that develops Selenoid) provides Moon, an as-a-service web browser service. The number of contracted web browsers in Moon are the slots of the service, being individual browsers that can be used in parallel to test the applications.

3. RETORCH* cost estimation

This section describes how RETORCH* supports the estimation of the cost implied by the execution of the E2E test cases in the Cloud, which we refer to as the *Overall cost*. From the RETORCH* model we get the duration of different lifecycle phases, and the Contracted and Minimal Capacities used during these phases. However, to link the Capacities used/Contracted, the time with the estimation of the different costs, we need three new concepts: the **Utilization**, the **Overprovisioning**, and the **Usage Profile:**

- **Utilization**: the percentage of the total Contracted Capacity of a Cloud Object Instance that is used by a single TJob in a concrete COI and TJob lifecycle phase (Set-up, Test Execution, or Tear-down). The calculation of the utilization of the total Contracted Capacities is based on the division against the TJob Resources Minimal Capacities multiplied times 100.
- **Overprovisioning**: the percentage of Contracted Capacity and time of a COI that is not used by the Execution Plan, resulting in an idle infrastructure and impacting negatively in the efficient use of the Contracted Cloud Object Instance.
- **Usage Profile**: graphical representation of the utilization of Contracted Capacities across different lifecycle stages of multiple Cloud Object Instances within the same Cloud Object category during the time that are provisioned. The Usage Profile shows the utilization of Contracted Capacities at each COI and TJob lifecycle stage, as well as the overprovisioning during the provisioned time.

Fig. 5 presents a simple example of a Usage Profile of Service (COI) with a Contracted Capacity of 4 slots (shown in the vertical axis), under a pre-invoiced Billing Option with a two-minute *billing period* (120 s shown in the horizontal axis). We use this example to outline how we apply the model estimations.

The COI takes 15 s to Set-up (shown in grey in Fig. 5), after which the TJob Execution starts. The first three TJobs (TJob 1, 2, and 3) require

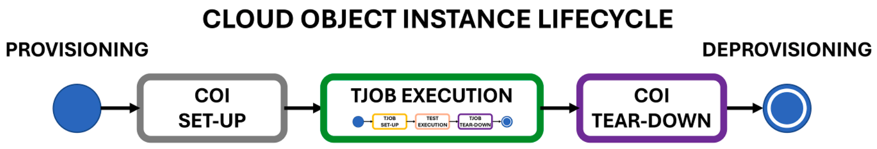


Fig. 4. COI lifecycle phases.

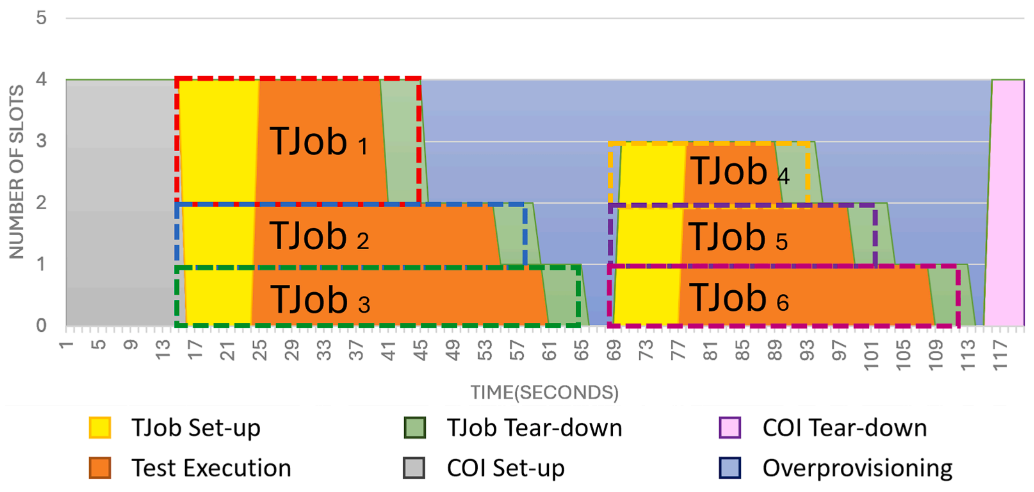


Fig. 5. Example of an as-a-service web browsers Usage Profile.

2,1 and 1 slots of Contracted Capacities (vertical axis). The three TJobs start their execution at the 16th second, taking 8 s to complete their Set-up (yellow in Fig. 5), 16, 30, and 35 s of the TJob execution (shown in orange in Fig. 5) and 5 seconds the Tear-down (shown in green in Fig. 5). The three first TJobs are followed by the next ones (TJobs 4, 5, and 6), which start their execution at the 70th second. These TJobs also take 8 s for the Set-up, 12, 22 and 30 s respectively for the Test Execution and 4 s to Tear-down. Last, the COI enters its tear-down, which takes 4 s (shown in violet in Fig. 5). The Fig. 5 blue area that remains beyond the rest of the colors represents the overprovisioning (i.e., slots of Contracted Capacities not used by any TJob).

The Usage Profile provides insights about the utilization for all lifecycle phases of both the infrastructure and the TJobs that would run into the COI as well as the overprovisioning. This utilization can be calculated by measuring the areas contracted and used by the different TJobs-COI phases. In this example, we have 480 slots-seconds available (120 s x 4 slots provisioned). Of these, 60 slots-seconds are allocated to the COI setup, 60 slots-seconds to the TJobs Set-up, 67 slots-seconds to the Test Execution, 34 slots-seconds to the TJob Tear-down, and 20 slots-seconds to the COI Tear-down.

In utilization terms, means that the COI utilizes 12.5% for the Set-up and 4.1% for the Tear-down while the TJobs utilize 12.5% for Set-up, 13.95% for the Test Execution, and 7.08% for Tear-down. The remaining provisioned slots-second (143) represent an overprovisioning of 29.79%.

After this illustrative example, we can summarize the definition of these costs as follows:

- **Overall cost:** the cost that is invoiced by the Cloud Provider for the Cloud Infrastructure contracted, calculated as the product between the Cloud Object Instance Contracted Capacities Invoiced Price and the contracted time. The Overall cost can be distributed over several executions of the same or different Execution Plans that use the Cloud Infrastructure.
- **Testing cost:** the cost of Contracted Capacities and time employed in tasks related to the testing process during the Execution Plan execution. The testing costs are computed considering the lifecycles of both TJobs and Cloud Object Instances:
  - **Execution cost** represented by the orange area in Fig. 5, is the cost associated with TJobs Test Execution lifecycle phase.
  - **Set-up cost** represented by the gray and yellow areas in Fig. 5, is the cost of the COI and TJobs Set-up lifecycle phases.
  - **Tear-down cost** represented by the violet and green areas in Fig. 5, is the cost of the COI and TJobs Tear-down lifecycle phases.
- **Overprovisioning cost** represented in blue color in Fig. 5, is the cost of the rest of the Contracted Capacity that is not used for testing tasks.

4. Evaluation methodology design

In this section, we provide the formulation design of the study we conducted to evaluate our cost model. Precisely, according to the critical analysis by Wohlin and Rainer (Wohlin and Rainer, 2022), our study can be qualified as a *small-scale evaluation*. In Section 4.1 we define the Research Questions, and in Section 4.2 we present the context of our evaluation (including the System Under Test, the infrastructure and the different assumptions). Then Sections 4.3 to 4.6 present how the evaluation is carried out, focusing on the four major decisions that impact the cost and efficiency of test execution to the Cloud: I) the Cloud Objects; II) the Billing Option; III) the Execution Plan of TJobs; and IV) sharing the infrastructure among different projects.

4.1. Research Questions

To derive our Research Questions we employ the *Goal Question Metric (GCM)* paradigm (Basili et al., 1994). The **goal** of the study is evaluating how RETORCH* can support the test and Cloud configuration when E2E testing is carried over the Cloud. We address the *purpose* of assessing the impact of three main selection decisions: a) the Cloud Objects, b) the billing option, and c) the Execution Plan. The *perspective* is from the tester point of view, who is interested in evaluating the impact of those decisions on the costs of E2E testing in the Cloud. Based on the three above decisions, we identify the three following **Research Questions**:

RQ 1: Can RETORCH* support the tester decisions by comparing how the costs of the different test and Cloud configurations vary by:

RQ 1.a: Choosing different Cloud Object categories?
RQ 1.b: Choosing different Billing Options?
RQ 1.c: Setting different Execution Plans?

The same **goal** could also be pursued from the same *perspective* with a different *purpose*: assessing how the costs vary when a common Cloud infrastructure is shared among several projects and teams. We then ask an additional **Research Question** as follows:

RQ 2: Can RETORCH* support the tester decisions by comparing how the costs of the different Cloud and test configuration vary when the infrastructure is shared among different projects?

To answer the above four Research Questions, we observe RETORCH* outcomes by assessing the following three **metrics**:

M1: The costs provided by the RETORCH* model, measured in dollars: Overall cost, Test Execution cost, Set-up cost, Tear-down cost and the Overprovisioning cost.
M2: % of Utilization
M3: % of Overprovisioning

Concerning M2 and M3, in addition to the metrics we also rely on the graphical representation of the **Usage Profile**, which is a key tool provided to the testers for aiding their decisions. Indeed, our assumption is that these metrics and the Usage Profile can be inspected by the tester as a support for deciding which test and Cloud configuration is the most convenient for the system test process on hand.

In our evaluation study we employ the following strategy: given the four decisions that impact the cost of E2E Cloud testing (Cloud Object categories, Billing Options, Execution Plans and Infrastructure sharing), in each RQ we vary only one of them while fixing the other ones. Each RQ is answered in an individual subsection, in which we provide its individual set up, the evaluation results and its analysis. Precisely:

- To address **RQ 1.a** in Section 4.3, we set the Execution Plan obtained as output from the RETORCH orchestration tool (Augusto et al., 2020) and Pay-as-you-go Billing Option type, and we explore the selection of different Cloud Object *categories.*
- To address **RQ 1.b** in Section 4.4, we set the Execution Plan and the Cloud Objects explored in **RQ 1.a** and we vary the Billing Option *types* by comparing the Pay-as-you-go option (the same as **RQ 1.a**) with the Pre-invoiced one.
- To address **RQ 1.c** in Section 4.5, we fix the Cloud Object *categories* and Billing Option used in **RQ 1.a** and we use different Execution Plans which vary the TJob parallelism.
- Finally, to address **RQ 2** in Section 4.6, we use as Cloud Object category the virtual machine, and the Execution Plan and Billing option used in **RQ 1.a**. We increase the capacities of the virtual machine, albeit one that is already in use by other Execution Plans to explore the impact of sharing the test infrastructure.

4.2. *Context of the evaluation*

As the **subject of the evaluation**, we use a real-world application called *FullTeaching* (ElasTest EU Project 2017), a demonstrator in the European Project ElasTest (Garcia et al., 2018). The application provides a platform to ease online teaching with videocalls, forums or messaging, and uses complex Resources such as web and multimedia servers or databases, as well as Resources required during the testing phase as web-browsers. The application has available two test suites in the ElasTest-URJC repositories (ElasTest EU Project 2017), with a total of 21 E2E test cases, which validate the teacher and students functionalities, e. g., by creating a new course, posting comments, attending a classroom, sending messages. This test suite was the input to the RETORCH tool, that provides the Execution Plan with the different TJobs in sequential and/or in parallel way.

The Continuous Integration infrastructure used is a physical server running Windows Server 2019 with Hyper-V hypervisor, on which a Jenkins continuous integration environment with several agents is installed. The test runs used in the evaluation were executed in a dedicated agent with 12 virtual cores and 32 GB of memory of the 32 threads (2 $\times$ Xeon CPU E5–2620 v4 processors) and 128 GB of memory installed. The agent has access to the virtual machine *Docker* engine being able to deploy and tear down the different Resource Instances, through the commands of each TJob. The browsers are provided containerized by Selenoid as required by the TJobs, where each session is recorded and stored for later analysis. This infrastructure emulates the existing on-premises infrastructure that is intended to be migrated to the Cloud for executing the E2E test suite (see Section 4.3)

Dataset creation: using the above-described CI infrastructure, we executed 10 times each Execution Plan and collected the different times of its COI and TJob lifecycle phases. We have processed these 10 consecutive executions to create a dataset with the average duration each lifecycle (Augusto et al., 2024) (onwards referred to as the average datasets).

To estimate the different costs, we use the average datasets of each Execution Plan with a series of parameters: the **number of Execution Plan executions/hour and the extrapolated time**, the **COI Contracted Capacities,** and the **estimated COI Set-up time.**

The **number of Execution Plan executions/hour** is set to 3: this number has been set as a plausible value, after analyzing the peak of executions (4) of our CI system. The **extrapolated time** used to analyze the RETORCH* costs is one year, as well as the minimal billing-period with discount available in the Cloud provider used for the **RQ 1.b** containers and virtual machines, see below. The **Contracted Capacities** and the **estimated COI Set-up time** are set according to the different Cloud Object categories as the following:

- **Virtual machines:** are supposed to have a similar Set-up time and Contracted Capacities of the on-premises agent (32 GB and 12 cores).
- **Containers:** the average Set-up time of the COI is increased in 1 min (Janakiram, 2023) and are contracted according to the Resource Minimal Capacities observed in our continuous integration system (e.g., the database has 0.29 GB of memory and 0.2 virtual processors).
- **Services:** are intended to be provisioned with no-Set-up time and including as many Contracted Capacities as required.

Cloud Provider: We have considered multiple providers to highlight the "*Cloud Provider agnosticism*" of our model and to explore various billing options offered by these Cloud Providers. While most major Cloud Providers offer similar services at comparable prices, this decision allows us to delve into their nuances. In our case, we have chosen services from three leading Cloud Providers: *Google Cloud Platform, Azure,* and *AWS*.

For better readability, related to the Usage Profiles in the figures below we show the Contracted Capacities used during 1 hour of the one-year time extrapolated. In the case of the containers, the Contracted Capacities that are shown correspond to the entire container group (a set of containers contracted and invoiced together) to ease both the representation and analysis.

4.3. *[RQ 1.a]: Impact of the Cloud Object category*

To answer RQ 1.a, we analyze how RETORCH* supports the tester in selecting one Cloud Object *category* by showing it impact on the cost (**M1**), utilization (**M2**) and overprovisioning (**M3**) of the E2E test execution.

4.3.1. *Evaluation set-up*

To evaluate the impact of the Cloud Object *category*, we fixed both the Billing Option and the Execution Plan and explored three different Cloud Object *categories offered* by three major Cloud Providers. The Cloud Object Instances and their Billing Options were retrieved from *Google Cloud Platform* (Google, 2023), *Azure Container Instances* (Microsoft 2023), and *AWS Device Farm* (Amazon 2023c) portals on the 21 of June of 2023 and are the following:

1. **Virtual Machine** into the *Google Cloud Provider* (N1) on which we deploy all Resource Instances located in the Region of London (*Europe-west2*). The virtual machine Billing Option has a *billing period* of 1 hour with Contracted Capacities Invoiced Prices of \$0.05453/GB of memory and \$0.040692/vcore of processor.
2. **Containers** are provisioned in *Azure Container Instances*, deploying each Resource Instance into a separate container in the North Europe region. The containers Billing Option has a *billing period* of one

second and its Contracted Capacities Invoiced Price is \$0.0000013/GB and \$0.0000113/vcore.

3. **Services** to provide the web browsers combined with the rest of the Resource Instances deployed into containers. The services are provisioned by *AWS (Device farm*) and their Billing Option has a 1-minute *billing period* with a Contracted Capacity (slots) Invoiced Price of \$0.19/slot. The rest of Resources Instances are deployed into *Azure Container Instances* with identical Billing Option and region that the above item.

The **Execution Plan** is specified into a pipelining code (Jenkins file) generated by our RETORCH tool. The Execution Plan is depicted in Fig. 6 Ⓐ, the boxes represent the TJobs labeled with different letters (C to N). Each group of TJobs (e.g., C, D, E, F, and G) is parallelized and synchronized with the following group (e.g., TJobs C-G are synchronized with TJobs H-L) through different Jenkins Stages. The length of each TJob corresponds to its duration in seconds, and its positioning in the figure depicts when it is executed in the Execution Plan.

Fig. 6 Ⓑ represents the number of Resource Instances required during the Execution Plan execution: web servers (blue), OpenVidu servers (dark green), mocks of the OpenVidu server (light green), web browsers (red) and databases (orange).

Table 1 depicts the data contained in the average dataset of the RETORCH Execution Plan. The first column represents each TJob whereas the columns groups COI and TJob depict the average of durations (difference between the start and end) in seconds of each lifecycle phase, respectively.

4.3.2. Evaluation results

Fig. 7 depicts the Usage Profiles of the different Contracted Capacities the memory (Ⓐ), processor (Ⓑ), and slots (Ⓒ) for the virtual machine (①) deployed in *GPC (Google Cloud Platform)*, the container group (②) in *Azure* (*Container Instances*) and the devices with containers (③) contracted in *AWS* (*Device Farm*) and *Azure* respectively.

The Contracted Capacities used during the different lifecycles are represented in all graphs with the same colors: the yellow color Set-up of the COI/TJobs, the Test Execution is in orange, and finally, the Tear-down is represented in violet color (slight line at the end of each Test Execution phase). Additionally, the blue color represents the Contracted Capacity that is overprovisioned.

In the virtual machine (Fig. 7 ①) there is a significant over-provisioning (**M3**) of the two Contracted Capacities. The remaining Contracted Capacities utilization (**M2**) are allocated to the Set-up and Test Execution of the TJob, with a smaller portion allocated for the Tear-down of the TJobs-COI. The high percentage of overprovisioning (**M3**: 77% and 70% respectively) is due to a fixed one-hour *billing period* not allowing earlier release.

In the containers (Fig. 7 ②) and the services (Fig. 7 ③) alternatives,

Table 1
RETORCH Execution Plan times.

TJob	COI		TJob		
	Set-up (s)	Tear-down (s)	Set-up (s)	Execution (s)	Tear-down (s)
C			54	96.7	4.8
D			58.8	122.9	2.5
E			59.8	104.6	2.8
F			62.3	140.4	2.7
G			63.5	138.3	2.3
H	3.7	1.2	47.3	89	2.8
I			46.4	96.8	2.5
J			50.4	297.8	2.3
K			50.6	154.1	2.1
L			53	86.9	2.8
M			29.6	59	2.6
O			30.7	53.5	2.4

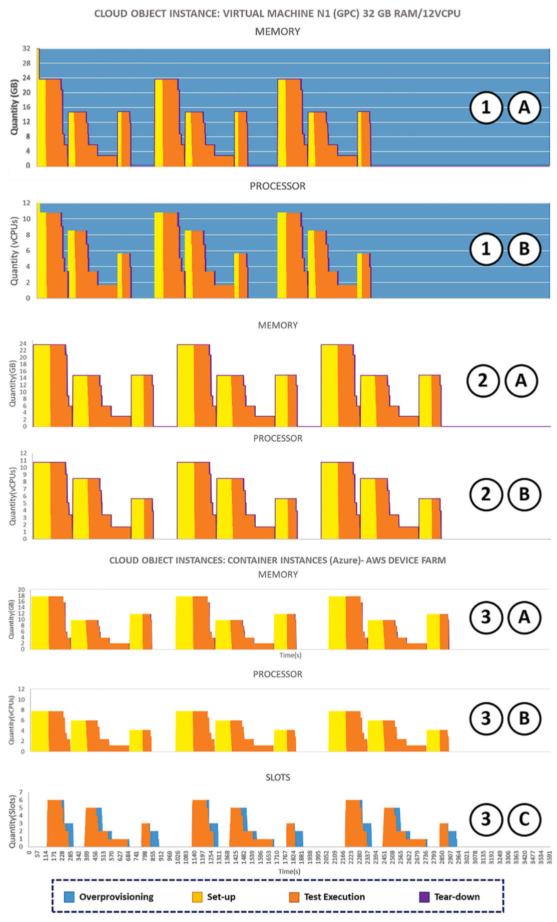


Fig. 7. Usage Profile Virtual Machine, Containers, and Services.

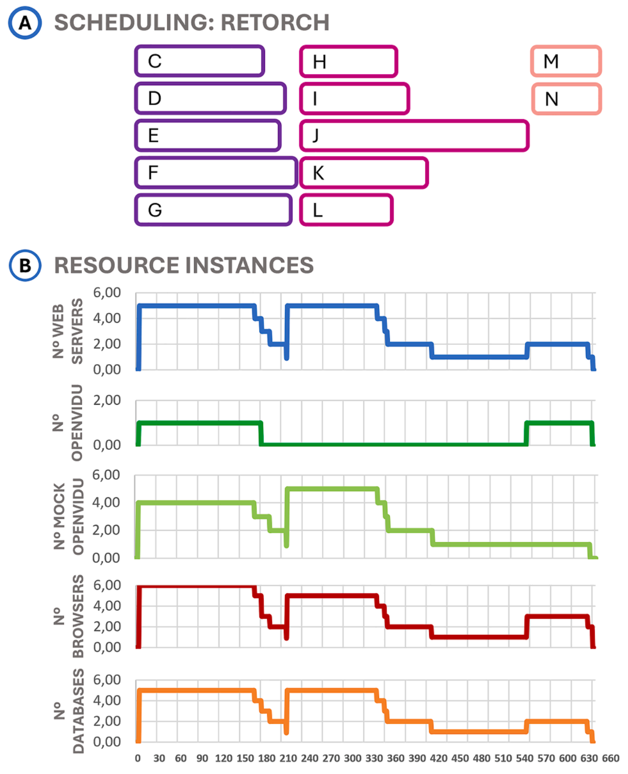


Fig. 6. RETORCH Execution Plan.

we can observe a similar percentage of usage (**M2**) in memory and processor, lower in the containers because the browsers are not containerized. The containers show a zero percentage of overprovisioning (**M3**) due to the short *billing period* of the Billing Option (1 second) enabling the release of the COI at the same moment that the TJob ends. On the other hand, the service slots used to deploy the browsers in Fig. 7 (③ Ⓒ) present the opposite case where the Contracted Capacity presents a percentage of overprovisioning (**M3**) of16% by its longer *billing period* (1 min).

In both containers and services, the percentage of utilization (**M2**) of the different Contracted Capacities and costs (**M1**) are primarily allocated to COI/TJob Set-up and Execution, and a smaller portion is dedicated to Tear-down; on the other hand, the services (slots) are primarily allocated to the TJob/COI Execution with no Contracted Capacities used for the Set-up and Tear-down.

Table 2 shows for each COI the percentage of utilization(**M2**), overprovisioning (**M3**) and the different costs (**M1**) of the Contracted Capacities (in dollars) for each lifecycle of the COI/TJob phase (Set-up, Execution, and Tear-down) as well as its Overall cost.

4.3.3. *Evaluation analysis*

Considering the Overall cost (**M1**), the containers with a total cost of $2097.53 are 36% and 96% cheaper than the virtual machine and services respectively. Analyzing the alternatives in terms of testing cost (**M1**), the virtual machine with a cost of $1622.91 is 23% and 96% cheaper than the containers and services (**M1**: $2097.53 and $37,992.93). The services are significantly more expensive (over 90% in terms of Overall cost (**M1**)), due to the browser service price, but shows how the Resources cost can be invested only in the TJob execution with no Set-up or Tear-down costs.

Given the above results, we could answer the RQ 1.a as:

RQ 1.a: RETORCH* enables the tester to compare the different Cloud Object *categories:* Containers are more attractive in terms of Overall cost (**M1**) and no Overprovisioning (**M3**), while the virtual machines present a better Testing cost, which is the sum of Set-up, Execution and Tear-down (see Section 3) (**M1**) but a high percentage of Overprovisioning (**M3**). Finally, in the Services the cost of Slots is entirely invested in Test Execution cost but Overall (**M1**) is by far the most expensive option.

4.4. *[RQ 1.b] Impact of the Billing Option*

To answer RQ 1.b, we aim to analyze how RETORCH* could supports the tester decision of selecting among different Billing Option *types* by showing its impact on the cost (**M1**), utilization (**M2**) and overprovisioning (**M3**) of E2E test execution. It is common among most Cloud Providers to offer Cloud Object Instances with several Billing Options, primarily differing in the pricing for the Contracted Capacities, depending on the amount of time that is contracted (*billing period*).

4.4.1. *Evaluation setup*

To conduct the evaluation, we have fixed the Cloud Object categories and Execution Plan as those selected in RQ 1.a. We compare the As-you-go Billing Options type studied in RQ 1.a with the pre-invoiced Billing Option offered by the same Cloud Providers, projected over one year (the shortest period available for the pre-invoiced option). The Billing Option types, Invoiced Prices, and *billing periods* considered are summarized in Table 3.

4.4.2. *Evaluation results*

Fig. 8 illustrates the Overall costs (**M1**) of the Virtual Machine (Ⓐ), containers (Ⓑ), and services (Ⓒ) for both Billing Option *types*: As-you-go and Pre-invoiced. The x-axis represents the time that the Cloud Object Instance is provisioned in hours, ranging up to 8760 h (equivalent to 1 year), while the y-axis represents the Overall cost (**M1**) of the three Billing Option *types*. The three graphs show a trade-off point at which the Overall cost (**M1**) of the As-you-go *type* exceeds the Pre-invoiced.

This trade-off point is reached at the 63% of usage (**M2**, 5519th hour) in the case of the Virtual Machine (Fig. 8 Ⓐ), 73% of usage (**M2**, 6395th hour) in the containers (Fig. 8 Ⓑ), and 33% of usage (**M2**, 2900th hour) in the services (Fig. 8 Ⓒ). Table 4 shows for each Cloud Object *category* the percentage of utilization (**M2**) and the different costs (**M1**) of the Contracted Capacities (in dollars) for each lifecycle phases of the COI/TJob phase (Set-up, Execution, and Tear-down) as well as its Overall cost (**M1**) and its Billing Option *type* (As-you-go and Pre-invoiced):

4.4.3. *Evaluation analysis*

Our findings using RETORCH* show that, as expected, the selected type of Billing Option directly impacts in the percentage of overprovisioning (**M3**) and percentage of utilization (**M2**) of the Cloud Object Instances. In terms of costs (**M1**), after a trade-off point, the as-you-go *type* becomes more costly than the pre-invoiced, which becomes the most convenient contract. However, we are not considering here the possible percentage of overprovisioning (**M3**) of the Contracted Capacities. Using as reference the cost invested with a Pay-as-you-go, the virtual machine is more cost-attractive up to $3657.76 invested, the containers up $1127.50 invested, and the services up to $14,400.00 invested.

Table 2
RQ1.1. Costs and utilization of the Cloud Object Instances.

COI	Contracted Capacity	M2–3: Utilization (%)				M1: Costs ($)				
		Set-up	**Test Execution**	**Tear-down**	**Overprovisioning**	**Overall**	**Set-up**	**Test Execution**	**Tear-down**	**Overprovisioning**
Virtual Machine	Memory	5%	15%	1%	79%	$1528.6	$111.5	$234.2	$7.4	$1175.4
	Processor	9%	20%	1%	70%	$4277.5	$390.1	$855.3	$24.4	$3007.8
TOTAL COSTS						**$5806.1**	**$501.6**	**$1089.5**	**$31.8**	**$4183.2**
Containers	Memory	49%	49%	2%	0%	$405.9	$198.5	$201.0	$6.4	$0.0
	Processor	48%	50%	2%	0%	$1691.6	$812.3	$855.0	$24.4	$0.0
TOTAL COSTS						**$2097.5**	**$1010.8**	**$1056.0**	**$30.7**	**$0.0**
Services	Memory	49%	49%	2%	0%	$295.3	$145.9	$145.1	$4.4	$0.0
	Processor	48%	50%	2%	0%	$1212.2	$583.4	$611.9	$16.8	$0.0
	Slots	0%	84%	0%	16%	$43,484.6	$0.0	$36,485.4	$0.0	$6999.2
TOTAL COSTS						**$44,992.2**	**$729.3**	**$37,242.4**	**$21.2**	**$6999.2**

Because of the abovementioned, we could answer RQ 1.b as:

RQ 1.b: The costs calculated by RETORCH* enable testers to compare the different Billing options and help them devising which is the most attractive option. In terms of the Overall cost (**M1**), RETORCH* shows that the threshold beyond which the pre-invoiced type should be selected.

Table 3
Billing Option types, Invoiced prices, and Billing periods.

COI	Billing Option Type	Invoiced Price	Billing periods
Virtual Machine	As-you-go	$0.135/GB $0.33/vcore	1 h
	Pre-invoiced	$0,0837/GB $0.2046/vcore	1 year
Containers	As-you-go	$ 0.0000013/GB $0.0000113/vcore	1 s
	Pre-invoiced	$0.000000949/GB $0.000008249/vcore	1 year
Services	As-you-go	$0.19/slot	1 min
	Pre-invoiced	$200/slot	1 month

4.5. [RQ 1.c] Impact of the Execution Plan

To answer this research question, we analyze how RETORCH* supports the tester in comparing different Execution Plans in terms of cost (**M1**) and percentages of utilization (**M2**) and overprovisioning (**M3**). Execution Plans can vary in terms of execution time and Contracted Capacities required (resulting in higher or lower parallelism). We analyze two Execution Plans that reduce the parallelism among TJobs in comparison with the original RETORCH Execution Plan (reducing from 5 to 4 and 3 threads of parallelism). The study is done with the Cloud Object categories used in RQ 1.a, and considering the As-you-go Billing Option type.

4.5.1. Evaluation SET-UP

Fig. 9 depicts the 4-Parallel and 3-Parallel Execution Plans as well as the different Resources required:

Table 5 depicts the different TJob and COI durations, collected from our CI system for the 4-Parallel and 3-Parallel Execution Plans.

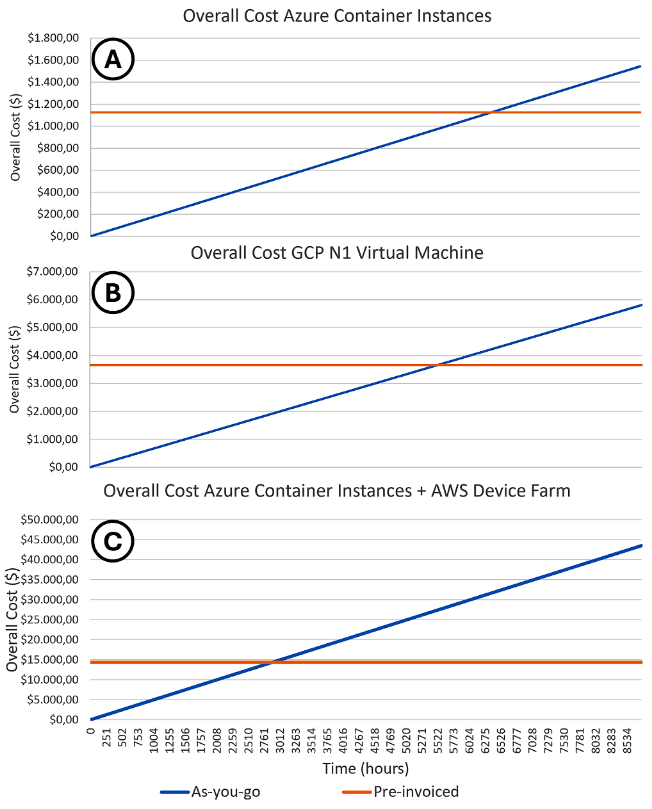


Fig. 8. Overall Costs of the virtual machine.

4.5.2. Evaluation results

Fig. 10 illustrates the Usage Profile of the virtual machine (①, ④), containers (②, ⑤), and the combined usage of containers and *AWS* device browsers as a service (③, ⑥) for both Execution Plans (4-Parallel and 3-Parallel). These profiles showcase the percentages of utilization (**M2**) and overprovisioning (**M3**) for the different Contracted Capacities: memory (Ⓐ), processors (Ⓑ), and slots (Ⓒ). Once again, the colors differentiate the Contracted Capacities used: yellow color represents Set-up phase, orange represents the COI/TJob Execution phase, violet represents the COI/TJOB Tear-down phase, and blue the overprovisioned.

Considering the Usage Profiles for each Execution Plan and Cloud Object Instance, we employed RETORCH* to estimate the costs (**M1**), percentage of utilization (**M2**) and overprovisioning (**M3**) and compare them to the original RETORCH Execution Plan. Table 6 provides a summary of the results, the first two columns specify the Cloud Object Instances and Execution Plan while the next columns present the costs (**M1**):

In the three Execution Plans, reducing the parallelism of the Execution Plan results in a decrease in the cost (**M1**) of COI/TJobs Set-Up, Execution, and Tear-down in the virtual machine: $501.63 to $446.84 to $400.48 the set-up (**M1**), $1089.48 to $1008.07 to $918.19 the execution (**M1**), and $31.80 to $30.16 to $30.05 the Tear-down (**M1**). However, the cost reduction is offset by an increase in overprovisioning costs (**M1**), which rise from $4183.22 to $4321.06 to $4456.41.

Similarly, the containers also experience reduced COI/TJobs Set-up cost (**M1**: $1010.76, $999.54, and $915.87), Execution cost (**M1**: $1056.02, $1014.77, and $888.83), and Tear-down cost (**M1**) ($30.74 to $30.24 to $30.07) costs without Overprovisioning cost (**M1**). Nevertheless, the container alternative remains the cheapest Cloud Object Instance for executing the Execution Plan, with costs (**M1**) of $2097.53, $2044.55, and $1834.25.

Furthermore, the combined use of services and containers yields different prices depending on the COI/TJobs Set-up cost (**M1**: $729.30 to $675.15 to $673.36), Execution cost (**M1**: $37,242.43 to $35,276.44 to $23,353.11), and Tear-down cost (**M1**: $21.19 to $19.98 to $21.19) involving browsers.

4.5.3. Evaluation analysis

According to Amdahl's law (Gene and Amdahl, 1967), which defines the theoretical speedup of a task when parallelism is increased, our Execution Plan's execution time is expected to increase by 20% and 40% when reducing parallelism to 4 and 3 TJobs in parallel. However, in practice, the Execution Plan takes 659.8, 660.6, and 759.9 s (COI

Table 4
RQ 1.b Alternatives costs.

COI	Billing Option	Capacity	M2–3: Utilization (%)				M1: Costs ($)				
			Set-up	**Test Execution**	**Tear-down**	**Overprovisioning**	**Overall**	**Set-up**	**Test Execution**	**Tear-down**	**Overprovisioning**
VM	As-you-go	Memory	10%	16%	1%	73%	$1528.6	$156.9	$250.9	$8.1	$1112.6
		Processor	13%	20%	1%	66%	$4277.5	$530.3	$908.7	$27.7	$2810.8
TOTAL COST							**$5806.1**	**$687.2**	**$1159.6**	**$35.9**	**$3923.5**
VM	Pre-invoiced	Memory	10%	16%	1%	73%	$962.9	$98.9	$158.0	$5.1	$700.9
		Processor	12%	20%	1%	66%	$2694.9	$276.7	$442.3	$14.3	$1961.6
TOTAL COST							**$3657.8**	**$375.5**	**$600.3**	**$19.5**	**$2662.4**
Containers	As-you-go	Memory	49%	49%	2%	0%	$405.9	$198.5	$201.0	$6.4	$0.0
		Processor	48%	50%	2%	0%	$1691.6	$812.3	$855.0	$24.4	$0.0
TOTAL COST							**$2097.5**	**$1010.8**	**$1056.0**	**$30.7**	**$0.0**
Containers	Pre-invoiced	Memory	8%	9%	1%	82%	$216.5	$18.2	$18.4	$0.6	$179.3
		Processor	8%	9%	1%	82%	$911.0	$79.4	$83.6	$2.4	$745.7
TOTAL COST							**$1127.5**	**$97.6**	**$102.0**	**$3.0**	**$925.0**
Service	As-you-go	Slots	0%	84%	0%	16%	$43,484.6	$0.0	$36,485.4	$0.0	$6999.2
	Pre-invoiced		0%	23%	0%	77%	**$14,400.0**	$0.0	$3274.0	$0.0	$11,126.0

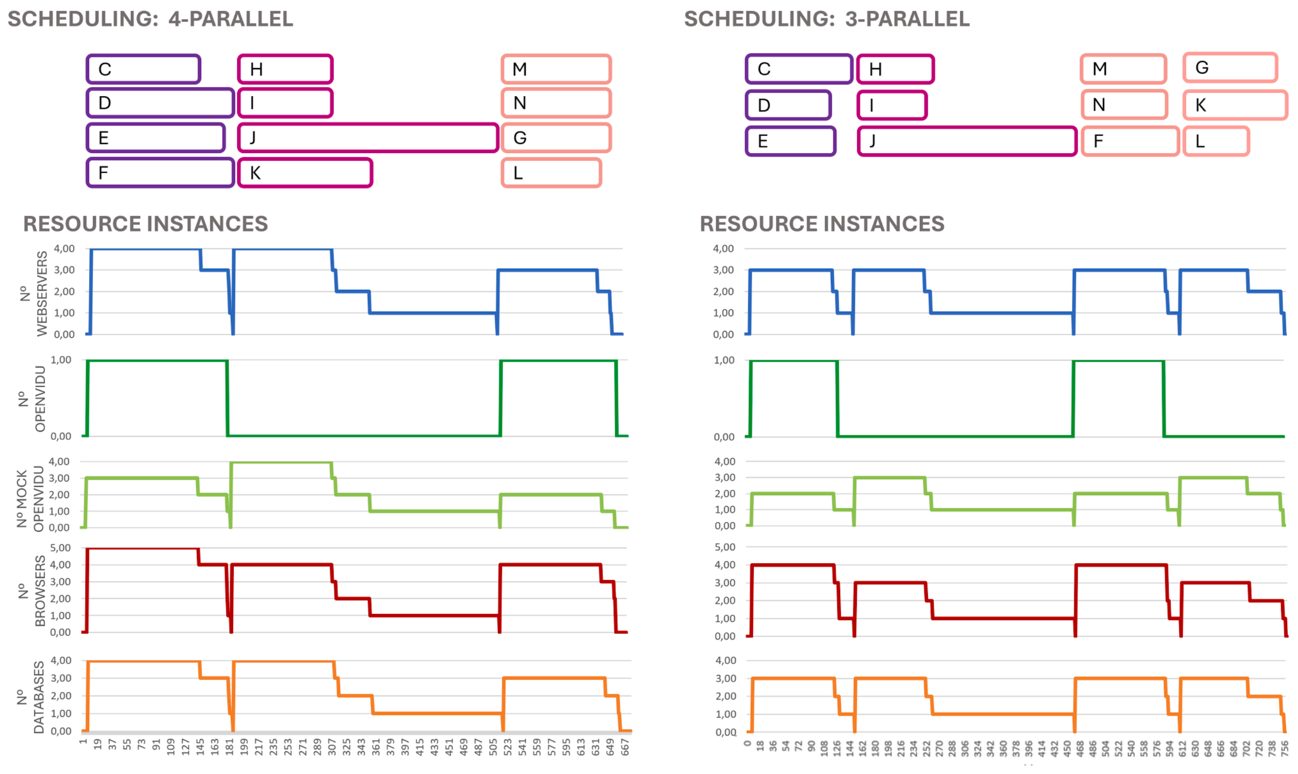


Fig. 9. 4-Parallel and 3-Parallel Execution Plans.

Tear-down from Table 1, Table 5, and Table 6). The transition from RETORCH to 4-Parallel shows no significant increase in the average time, while the 3-Parallel approach leads to only a 15% time increase instead of the expected 40%. Consequently, the required Contracted Capacities of Cloud Object Instances are lower, resulting in reduced costs (**M1**) and shorter execution times.

Considering the above-presented Usage Profiles, both Execution Plans exhibit higher percentage of overprovisioning (**M3**) in the virtual machine, due to the reduction of parallelism whereas the virtual machine Contracted Capacities remain constant. The containers Usage Profiles are similar, characterized by minimal percentage of overprovisioning (**M3**) and lower capacity requirements, due to less Contracted Capacity-intensive Execution Plans. The services contracted in the AWS device farm display a slight percentage of overprovisioning (**M3**) in Contracted Capacities, which can be attributed to their longer *billing period* of 1 min.

Given the abovementioned, we can answer RQ 1.c as follows:

RQ 1.c: The Execution Plan impacts the Cloud Object Instances utilization, mostly due to the parallelization level of the TJobs. Decreased parallelism of the TJobs leads to different capacities and time required, but also different Set-up, Execution, Tear-down and Overprovisioning costs. Consequently, a Cloud Infrastructure that was aligned with the tester's objectives in the past may no longer retain its optimal performance if the Execution Plan is changed.

Table 5
RQ1.c. 4-Parallel and 3-Parallel Execution Plan times.

TJob	Execution Plan	COI		TJob		
		Set-up (s)	Tear-down (s)	Set-up (s)	Execution (s)	Tear-down (s)
C	4-Parallel	3.9	1.1	40.5	79.3	3.1
D				42.1	91.7	2.3
E				45.1	77.3	2.6
F				44.8	106.3	2.2
G				36.9	71.1	2.4
H				39.7	80.2	2.1
I				41.1	281.5	2.3
J				42.3	133.7	2.4
K				45.5	121.5	2.4
L				46.8	83.4	2.8
M				42.5	100.4	2.3
O				44.4	82.9	3.2
C	3-Parallel	3.6	1	41.1	70.4	2.4
D				43	78	2.4
E				43.5	65.4	2.3
F				31.9	53.8	2.1
G				32.8	56.4	2.1
H				34.3	269.3	2.4
I				37.9	103.2	2.5
J				36.6	76.1	2.6
K				36.2	66.4	2.4
L				36.4	103.2	2.2
M				37.7	131.4	2.2
O				39.9	62.8	2.5

4.6. *[RQ 2] Impact of sharing*

To answer RQ 2, we analyze how RETORCH* supports the tester decision about sharing the Cloud Infrastructure by showing its impact on the cost (**M1**), utilization (**M2**) and overprovisioning (**M3**) of E2E test execution. It is typical for many companies to possess multiple applications and an already contracted Cloud Infrastructure, used for multiple purposes (e.g., testing, deploy in production, or pre-production tasks). This infrastructure might exhibit overprovisioned capacities that could be repurposed for testing. To address RQ 2 we analyze existing Cloud Infrastructure, comprising a Virtual Machine shared among multiple projects. We aim to deploy the subject application (*FullTeaching*) on this infrastructure to evaluate the cost (**M1**), utilization (**M2**) and overprovisioning (**M3**) of this deployment.

4.6.1. *Evaluation set-up*

To conduct this research question, we have fixed the RETORCH Execution Plan (Fig. 5) and the As-you-go Billing option type used in RQ 1.a The contracted Cloud Infrastructure consists of a N1 Virtual Machine procured through the Cloud Provider GPC with the Contracted Capacities increased to 96GB of memory and 48 vcores. This contracted Cloud Infrastructure is shared with other projects throughout the time that the COI is provisioned. This shared use of the infrastructure is simulated among more test suite executions of the FullTeaching test suite, using the collected data of the RETORCH Execution Plan with different numbers of executions per hour: the first two projects have 4 executions/hour whereas the third has 3 executions/hour.

4.6.2. *Evaluation results*

Fig. 11 depicts the Usage Profile of the Virtual Machine (precisely of

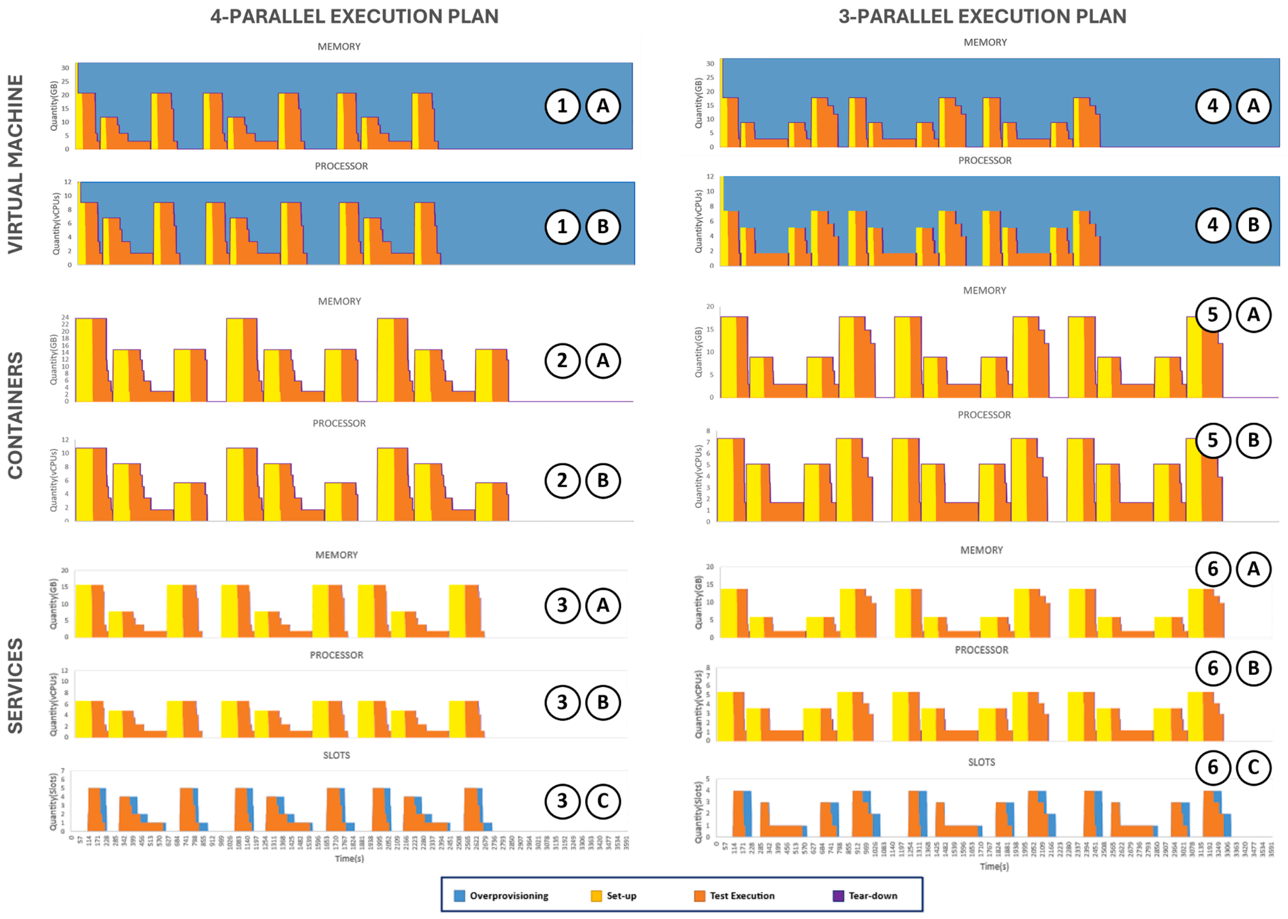


Fig. 10. Usage Profile VM, Containers, and Services with a 4-Parallel and 3-Parallel Execution Plan.

Table 6
Results RQ 1.c.

COI	Execution Plan	M1: Costs ($)				
		Overall	Set-up	Test Execution	Tear-down	Overprovisioning
Virtual Machine	RETORCH	$5,806.1	$501.6	$1,089.5	$31.8	$4,183.2
Containers		$2,097.5	$1,010.8	$1,056.0	$30.7	$0.0
Services		$44,992.2	$729.3	$37,242.4	$21.2	$6,999.2
Virtual Machines	4-Parallel	$5,806.1	$446.8	$1,008.1	$30.2	$4,321.1
Containers		$2,044.6	$999.5	$1,014.8	$30.2	$0.0
Services		$43,740.2	$675.1	$35,276.4	$20.0	$7,768.7
Virtual Machines	3-Parallel	$5,805.1	$400.5	$918.2	$30.0	$4,456.4
Containers		$1,834.3	$915.9	$888.8	$29.5	$0.0
Services		$31,816.8	$673.4	$23,353.1	$18.0	$7,768.7

Memory in Ⓐ, and Processor in Ⓑ). In the Usage Profiles, we maintain the same color scheme for each lifecycle phase as in the previous experiments, with a slight modification: the capacities used by other projects are represented in barred color:

In terms of Overprovisioning cost (**M1**), the virtual machine invests more than half of the Overall cost (**M1**) in capacities not used during the testing task that can be reused for other purposes (e.g., extra Execution Plan executions or execution of another Execution Plans) that would lead to increasing the Overall cost. The Usage Profile shows this percentage of overprovisioning (**M3**, blue color) but also enables the tester to take more informed cost-conscious decisions selection.

4.6.3. *Evaluation analysis*

Given the all the information presented in the analysis section, the tester based on the information provided by RETORCH* could take several decisions:

One alternative could involve reducing the number of Contracted Capacities for the different projects. As depicted in Fig. 11, the Virtual Machine experiences usage peaks in both Contracted Capacities from second 228 to 342 or seconds or 1938 to second 1955. Addressing these peaks could be achieved by adjusting the timing of RETORCH Execution Plan executions, for instance, starting at the second 500. This adjustment would reduce the number of Used Capacities, allowing for provisioning a virtual machine with fewer Contracted Capacities, e.g. with 36 vcores and 40GB of Memory.

By leveraging the Usage Profile, testers can also make decisions regarding Resource Instance sharing across all projects or between executions to reduce the Execution, Set-up, or Tear-down costs. For instance, it might be possible to use a single instance of Selenoid with enough available browsers or share Resource Instances that remain unmodified during the execution (e.g., proxies, databases used exclusively for read operations, authentication systems). Such decisions could result in a reduction of Contracted Capacities used during Set-up, Test Execution, or Tear-down and their cost (**M1**).

Lastly, another feasible decision could involve utilizing the extra Contracted Capacities for additional projects. In this virtual machine, starting from the 2100th second and onward, the available Contracted Capacities not used could facilitate the deployment of other projects or allow for running the Execution Plan more times.

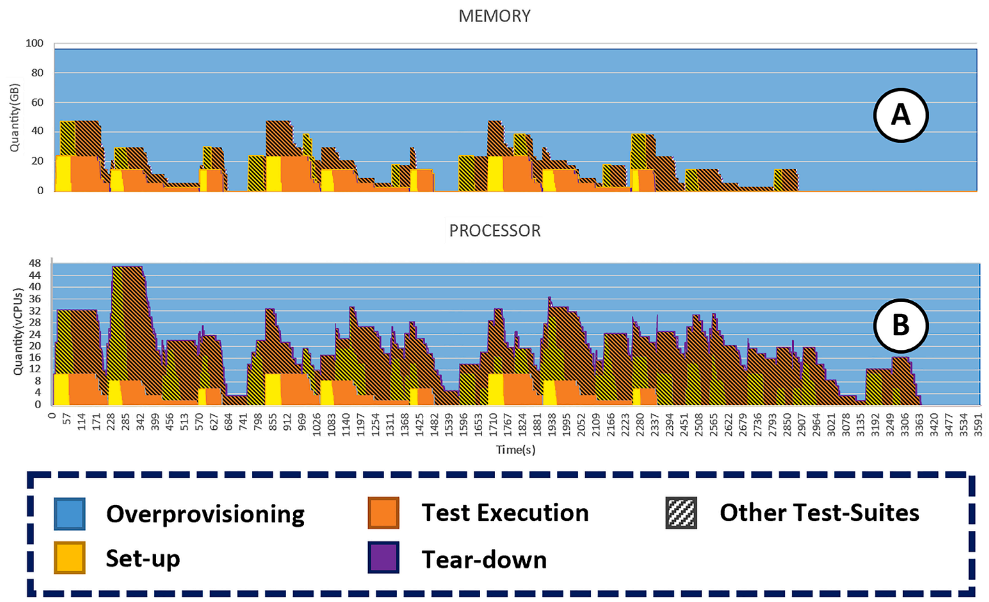


Fig. 11. Usage Profile shared infrastructure.

RQ 2: RETORCH* can support the tester decision of share the infrastructure as well as improve its efficiency during the E2E test execution. Based on the utilization, overprovisioning and the Usage Profile, the Test and Cloud Configurations can be tuned in order to share the use of an infrastructure. Based on the Usage Profile the shared infrastructure Test and Cloud Configurations can be tuned to improve the efficiency of the Execution Plan execution, such as sharing Resource Instances to avoid unnecessary redeployments, changing the Execution Plan arrangement to reduce the Contracted Capacity usage, or execute other projects in the same Cloud Infrastructure.

4.7. RETORCH* limitations

Despite the strengths of our model observed in the above study, we acknowledge a series of limitations thar could hinder the applicability of the model or the adoption by new practitioners/organizations. Those limitations are the following:

1. **Model Complexity and Usability:** The complexity of our model, while necessary to accurately capture the complexity of Cloud and test configurations, may present challenges for practitioners with a lack of knowledge in these areas. This could hinder the model's usability in real-world scenarios, concisely for developer teams without Cloud infrastructure experts.
2. **Dependence on Historical Data:** Our model relies on historical data for estimating costs and making informed predictions. While this approach is effective in scenarios where the historical data is available, it poses significant challenges in scenarios where the data is unavailable or has insufficient quality/information.

4.8. Threats to validity

Notwithstanding our diligent endeavors, the validity of the results for the different research questions above described remains susceptible to various threats. We acknowledge the existence of the following types (Wohlin et al., 2012):

4.8.1. Construct validity

Construct validity concerns the ability of a study to actually reflect the question investigated, in our case the assessment of costs for E2E testing in the Cloud. In our study, the construct validity could be impacted by our decision to measure the test execution time within an on-premises emulation of the test environment. While we did our best to properly reproduce the test environment as the one that would be acquired, our emulation could not be faithful to phenomena that may occur in the Cloud and thus we could introduce some error. To mitigate such issues, we paid particular attention to emulate the Cloud environment, and anyhow we consider that the same potential issues would possibly affect all the compared scenarios without introducing a bias.

4.8.2. Internal validity

One common concern regarding internal validity, i.e., potential causes that can affect the study conclusions, is the selection of the evaluation subjects. While it is true that the original subject chosen for our study has a limited number of test cases and lacks higher complexity elements like non-relational databases, proxies, and testing with mobile devices, it remains a real-world application that was utilized as a demonstrator in the European Project ElasTest (Bertolino et al., 2018). To address this potential limitation, we expanded the pool of test cases as much as possible by collecting them from multiple repositories (ElasTest EU Project 2017) resulting in a test suite size of 21 cases. By adopting this measure, we aimed to enhance the internal validity of our study, improve its robustness, and ensure a more comprehensive evaluation of the selected subject.

4.8.3. External validity

The generalizability and real-world applicability of the observed results from RETORCH* are subject to limitations due to various factors such as multiple Execution Plans, Cloud Objects under several Billing Options in different Cloud Providers. The extensive range of choices available in the Cloud, along with the various Execution Plans possibilities, makes it not feasible to test all possible combinations and directly affects the reliability and accuracy of our results. To ensure a comprehensive evaluation and reduce the impact of these external threats take specific measures: Firstly, we carefully select different *types* of Cloud Objects that represent the common services offered by the industry leaders. This approach enabled us to capture a more comprehensive understanding of RETORCH* utility by comparing different Cloud environments. Secondly, the selected Billing Options are the most representative: pay-as-you-go and subscription-based (Pre-invoiced). Third, we designed the different Execution Plans focused on the Cloud Object's usage impact, increasing and decreasing the parallelism of the execution.

Additionally, we acknowledge an external concern arising from the broad applicability of the RETORCH* model to different providers, options, and Cloud Objects. We recognize factors as the variation in Billing practices among different providers, such as Azure and *AWS*, which charge a 'fixed price' for their standard virtual machines, while *Google Cloud* adopts a more granular approach by invoicing each capacity separately. To address this concern, our concept of capacity provides flexibility to adapt to the evolving Cloud market and effectively address any change.

4.8.4. Reliability

To tackle this issue and ensure reproducibility by fellow researchers, we provide access to the user execution data, various configurations applied (C. Augusto et al., 2023), and the End-to-End (E2E) test suite used (C. Augusto et al., 2023).

5. Related work

This section analyzes the fields that are relevant to this article: Subsection 5.1 presents the state-of-the-art techniques and strategies to improve the efficiency of the test suite execution, Subsection 5.2 covers the testing in the Cloud, Subsection 5.3 deals with costs estimation when migrating to a Cloud infrastructure.

5.1. Efficient execution of test suites

The efficient execution of test suites remains an open challenge for testing (Bertolino, 2007). During the last decades, research trends have aimed at optimizing time and resources required to execute test suites. We classify these trends as (1) minimization/reduction/prioritization-based, (2) batching, and (3) test orchestration.

5.1.1. Minimization, reduction and prioritization techniques

The test prioritization, selection, and minimization (Yoo and Harman, 2012) techniques propose to order, select, or reduce the number of test cases achieving a similar defect detection rate at a lower time/cost; they have been validated in both academia and industry (Rothermel et al., 2002), (Wong et al., 1998). Test prioritization, selection, and minimization can be done according to several criteria e.g., dependency graphs to select those tests that cover the code that has changed, or likeness of failure according to the historical data (Greca et al., 2023).

Selection and minimization techniques (Yoo and Harman, 2012) are less effective with E2E test suites which are not usually composed of a huge number of test cases, as for unit or integration testing. Nevertheless, the complexity, duration, and resources required, still can require to identify a proper execution order or test subset. In this sense, the Execution Plan we derive using RETORCH is in fact aimed at prioritizing the tests to reduce test execution cost based on Resource usage; instead, most existing prioritization approaches use instead proxy parameters (such as coverage or similarity) for accelerating failure detection.

5.1.2. *Batching techniques*

Batching techniques are inspired by the strategies followed in the biomedical field to batch multiple repository changes (commits) and consist of executing the test suite at scheduled periods, rather than for each repository change. Despite its effectiveness only a few works have studied the impact of this type of techniques. Najafi et al. (Najafi et al., 2019) studied the batching impact in Ericsson finding reductions up to 42% in the execution time. Beheshtian et al. (Beheshtian et al., 2022) evaluated several batching techniques into a Travis CI, introducing several new batching approaches that considers a fix batch size, variate the size of batch according the load or risk among others. Bavand et al. (Bavand and Rigby, 2021) proposed to employ a dynamic-risk based batch size approach achieving time savings of 47%. Fallahzadeh et al. (Fallahzadeh et al., 2023) introduce the parallelism to the existent approach achieving reductions up to 81% in the number of machines required and execution reduction. Memon et all. proposed a strategy called TAP milestone (Memon et al., 2017), that bundle together consecutive code commits and execute/run the milestone as frequently as possible based on the available resources.

All the above batching approaches focus on reducing the number of executions in a continuous integration environment, trying to impact as less as possible the feedback time to the developer. On the other hand, our approach tries to support the tester to select the proper test and Cloud configuration to execute the E2E test cases (or the batches), so RETORCH* could be complemented to configure the Cloud infrastructure according to how the commits are batched.

5.1.3. *Orchestration techniques*

Orchestration techniques optimize the execution of test suites by ordering the test cases to reduce the execution time or the used resources. Most of the state-of-the-art techniques focus on increasing the level of parallelism. Chakraborty et al. (Chakraborty and Shah, 2011) propose to optimize the cost partitioning, grouping, and scheduling of test cases to parallelize them. Yu et al. (Yu et al., 2009), focus on discovering underlying dependencies through clustering techniques to enable the parallelization and share its resources without collateral effects. Cerny et al. (Frajtak and Cerny, 2022) propose an orchestration technique that seeks test case parallelization by resource sharing based on the type of database operations performed by the test cases. Finally, Garcia et al. (Garcia et al., 2018) propose an orchestration of the test suites based on the test execution (verdict-driven) or the output produced (data-driven).

Some of these techniques are effective also for optimizing E2E test case execution and show common aspects with our RETORCH* orchestration technique. Our resource characterization and access modes consider dependencies such as (Yu et al., 2009), we group and orchestrate test cases to reduce time costs as (Chakraborty and Shah, 2011), and we also use the type of operations performed by resources such as (Frajtak and Cerny, 2022). However, none of the above presented approaches has considered to select a proper Cloud Object combination and test orchestration, aligned with the tester objectives, as we do in RETORCH* to achieve a cost-effective E2E test execution in the Cloud.

5.2. *Testing over the Cloud*

In the testing field, there is a growing tendency of moving test suites to the Cloud (Bertolino et al., 2019) to achieve a better cost by leveraging unlimited and scalable resources, low entrance barriers, and avoiding part of the licensing costs or most of the infrastructure maintenance (Parveen and Tilley, 2010). Several authors have discussed the pros and cons of migrating testing to the Cloud (Parveen and Tilley, 2010, Khajeh-Hosseini et al., 2011). They warn that this migration requires to analyze carefully not only the SUT characteristics and the type of testing performed, but also new issues as legal, financial control, and software management (Riungu-Kalliosaari et al., 2012, Riungu et al., 2010)

Several works have shown that the Cloud is a cost-effective platform for concurrency and load testing (Inçki et al., 2012), and several frameworks have emerged to ease the testing in the Cloud. Yu et al. (Yu et al., 2010, Lian et al., 2009) introduced one of the first frameworks, ElasTest (Garcia et al., 2018) that also provides tools to execute the test suite in the Cloud or maximize the usage of the contracted Cloud Objects (Gambi et al., 2017). The industry response has been directed by big technological enterprises like CloudBuild (Microsoft) (Esfahani et al., 2016), Testin (Alibaba), Utest (Tecent), or MTC (Baidu) (Xie and Yang, 2018).

All these works share with RETORCH* the aim to achieve an efficient test execution when migrating the execution to the Cloud. However, previous works do not consider the optimization of the Cloud and the Test Configuration. The closest work to our proposal is the cost addon of ElasTest (Garcia et al., 2018), which estimates the Overall cost of the infrastructure used. Yet, to the best of our knowledge, most frameworks (Seybold and Domaschka, 2017) do not provide a fine grained cost estimation as the RETORCH* model. RETORCH* is not a Cloud framework itself, it is aimed at being used in combination with the other frameworks to estimate the different costs of the orchestration (test configuration) provided by these frameworks.

5.3. *Cloud service cost estimation*

Cloud Resource Orchestration techniques (Weerasiri et al., 2017) cover a broad range of approaches to allow users to escape from “the agony of choice” (Zhang et al., 2012) between the different services offered by the Cloud Providers. For the enterprises the objective is maximizing the revenue of their services/products, which lead to reduce their operational cost as much as possible. To measure the operational cost of a service/product the so-called Total Cost of Ownership (Li et al., 2009), which is implemented in most of the provider’s calculation tools (Google 2022, AWS 2022, Microsoft 2022), is generally used. Despite the utility of these calculators to estimate the cost, they are not interoperable between them, which hinders the comparison of multi-cloud or inter-provider solutions; in fact, third parties have created services and frameworks that enable the Cloud Provider and Instances comparison (Barnaby and Enykeev, 2022, Okraszewski, 2022). Garcia Galan et al. (García-Galán et al., 2017) propose a DSL to model the different requirements and a support of a tool for automating the selection of a Cloud infrastructure. Plewnia et al. (Plewnia, 2021) provide a tool that integrates several Cloud infrastructure optimization approaches. Abdennadher et al. (Abdennadher et al., 2017) propose a decision support system integrated with the Cloud Management platforms, which evaluates the cost of a customer’s applications using different service providers based on Cloud pricing and applications resource models. Truong and Dustdar (Truong and Dustdar, 2010) propose a service that enables the cost estimation, monitoring and analysis of scientific applications, to support the researcher in the decision about what parts of their application migrate to the Cloud. Zhang et al. (Zhang et al., 2012) present a prototype of a visual framework that allows the user to select and compare different Cloud services. Mezni et al. (Mezni and Abdeljaoued, 2018) present a Cloud service recommendation system based on Fuzzy formal analysis that can recommend different Cloud configurations based on the similarity to others users’ solutions.

RETORCH* differs from the abovementioned approaches (Zhang et al., 2012, Barnaby and Enykeev, 2022, Okraszewski, 2022, Plewnia, 2021, Abdennadher et al., 2017, Truong and Dustdar, 2010, Mezni and Abdeljaoued, 2018) in two key aspects. First, it focuses on achieving a cost-effective execution of the E2E test suites in the Cloud, while the other approaches are more generalist dealing with the migration/optimization of the existent Cloud Infrastructure. Second, while other approaches rely on solely the Overall cost, RETORCH* breaks down this cost in different sub-costs that enable the tester to make a more informed selection.

6. Conclusions and future work

We have presented RETORCH*, an E2E Test Execution Model in the Cloud, which represents the E2E test suite orchestration as well as the Cloud infrastructure and how it is used by the Execution Plan. RETORCH* can be used to compare different Cloud infrastructure configurations, considering not only the cost that is billed by the provider (overall cost) but also the cost incurred in the Execution Plan execution (testing cost) and the cost incurred in Cloud infrastructure that is not used (overprovisioning cost).

The RETORCH* model can be applied on several scenarios as a decision-support tool for the tester. One scenario is the usage of RETORCH * to support the migration of an existing test infrastructure to the Cloud, comparing different Cloud Objects and Billing options based on their different costs. Another alternative is using RETORCH* to improve a current Cloud or test configuration, comparing a new Execution Plans or Cloud Object to the one already used to deploy the test suite. Finally, another scenario is the use of RETORCH* for the continuous monitoring and improvement of a Cloud infrastructure, comparing the current infrastructure to new Cloud and test configurations to suggest changes that could lead to a more cost-effective execution (e.g. suggesting changes in the Execution Plan, or a new service offered by a provider to deploy as-a-service one Resource)

Through our evaluation study, we analyzed the practical application of RETORCH*. We are aware that the utility of our model cannot be adequately validated without a controlled experiment with human testers, which is known to require a huge amount of resources and time. This is planned in future work. In the context of this paper, our evaluation consisted of examining the impact of different costs provided by the model, including the use of different Cloud Objects, Billing Options, and Execution Plans. The evaluation highlighted the potential of RETORCH* for selecting the most suitable option and estimating the cost of executing the Execution Plan on existing contracted infrastructure. RETORCH* is conceived to empower testers to make informed decisions and choose alternatives that best align with their testing objectives.

In future work, we plan to continue the evaluation of RETORCH* with more real-world examples. We also plan to create a bot to enable the integration of RETORCH* into a CI system, to enable the continuous analysis and improvement of both the test infrastructure and the E2E test execution.

Declaration of generative AI and AI-assisted technologies in the writing process

During the preparation of this work, the authors used GPT-3.5/4-based correctors (e.g. Grammarly, Chat-GPT or Bing Copilot) to improve readability and language. After using these tools, the authors reviewed and edited the content as needed and take full responsibility for the content of the publication.

CRediT authorship contribution statement

Cristian Augusto: Conceptualization, Data curation, Formal analysis, Investigation, Methodology, Resources, Software, Visualization, Writing – original draft. **Jesús Morán:** Conceptualization, Formal analysis, Methodology, Supervision, Writing – original draft, Writing – review & editing. **Antonia Bertolino:** Conceptualization, Formal analysis, Methodology, Supervision, Writing – original draft, Writing – review & editing. **Claudio de la Riva:** Conceptualization, Funding acquisition, Methodology, Project administration, Software, Supervision, Writing – review & editing. **Javier Tuya:** Conceptualization, Funding acquisition, Project administration, Software, Supervision, Writing – review & editing.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Cristian Augusto reports financial support was provided by Spanish Ministry of Science and Innovation. Antonia Bertolino reports financial support was provided by Ministry of the Environment and Energy Safety. Jesus Moran reports financial support was provided by Spanish Ministry of Science and Innovation. Claudio de la Riva reports financial support was provided by Spanish Ministry of Science and Innovation. Javier Tuya reports financial support was provided by Spanish Ministry of Science and Innovation. Antonia Bertolino is a Senior Associate Editors of this Journal. If there are other authors, they declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The data/rp is: https://github.com/giis-uniovi/retorchx-rp.

Acknowledgments

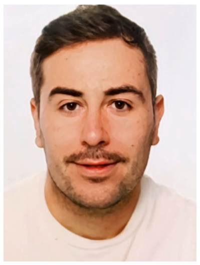
This work was supported in part by the project PID2019-105455GB-C32 under Grant MCIN/AEI/10.13039/501100011033 (Spain), in part by the project PID2022-137646OB-C32 under Grant MCIN/AEI/10.13039/501100011033/FEDER, UE, and in part by the project MASE RDS-PTR_22_24_P2.1 Cybersecurity (Italy).

References

Abdennadher, N., Loomis, C., Belli, O., 2017. An autonomic cloud application placement tool based on cost criteria. In: Proc. - 2017 IEEE Int. Conf. Cloud Auton. Comput. ICCAC 2017, pp. 143–152. https://doi.org/10.1109/ICCAC.2017.21.
Aliyun, “Alibaba cloud: reliable & secure cloud solutions to empower your global business.” Accessed: May 26, 2023. [Online]. Available: https://eu.alibabacloud.com/en.
Amazon, “Cloud Computing con Amazon Web Services.” Accessed: May 26, 2023. [Online]. Available: https://aws.amazon.com/es/what-is-aws/.
Amazon, “AWS Device Farm.” Accessed: Jun. 21, 2023. [Online]. Available: https://aws.amazon.com/en/device-farm/.
Augusto, C., Morán, J., Bertolino, A., de la Riva, C., Tuya, J., 2020. RETORCH: an approach for resource-aware orchestration of end-to-end test cases. Softw. Qual. J. 28 (3), 1147–1171. https://doi.org/10.1007/s11219-020-09505-2. Sep.
Augusto, C., Morán, J., Bertolino,A., de la Riva, C., and Tuya, J., “Replication package for ‘RETORCH: a cost and resource aware model for E2E Testing in the Cloud’.” Software Engineering Research Group (GIIS) of the University of Oviedo, 2023. [Online]. Available: https://github.com/giis-uniovi/retorchx-rp/.
Augusto, C., Morán, J., de la Riva, C., and Tuya, J., “FullTeaching E2E test suite.” 2023. [Online]. Available: https://github.com/giis-uniovi/retorch-st-fullteaching.
Augusto, C. Moran, J., De La Riva, C., and Tuya, J., “RETORCH* replication package: average datasets.” Accessed: May 06, 2024. [Online]. Available: https://github.com/giis-uniovi/retorchx-rp/tree/main/raw-datasets.
AWS, “AWS pricing calculator,” Aws. Accessed: Jun. 08, 2022. [Online]. Available: https://docs.aws.amazon.com/pricing-calculator/latest/.
Barnaby, J., and Enykeev, K., “Scalyr/cloud-costs.” Accessed: Jun. 08, 2022. [Online]. Available: https://github.com/scalyr/cloud-costs.
Basili, V.R. Caldiera, G., and Rombach, H.D., “The goal question metric approach,” *Encycl. Softw. Eng.*, vol. 2, pp. 528–532, 1994, https://doi.org/10.1.1.104.8626.

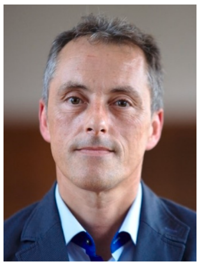
Bavand, A.H., Rigby, P.C., 2021. Mining historical test failures to dynamically batch tests to save CI resources. In: *Proc. - 2021 IEEE Int. Conf. Softw. Maint. Evol. ICSME 2021*, pp. 217–226. https://doi.org/10.1109/ICSME52107.2021.00026.
Beheshtian, M.J., Bavand, A.H., Rigby, P.C., 2022. Software batch testing to save build test resources and to reduce feedback time. IEEE Trans. Softw. Eng. 48 (8), 2784–2801. https://doi.org/10.1109/TSE.2021.3070269. Aug.
Bertolino, A., Calabró, A., De Angelis, G., Gallego, M., García, B., Gortázar, F., 2018. When the testing gets tough, the tough get ElasTest. In: Proceedings - International Conference on Software Engineering. ACM, pp. 17–20. https://doi.org/10.1145/3183440.3183497 in ICSE18.
Bertolino, A., et al., 2019. A systematic review on cloud testing. ACM Comput. Surv. 52 (5). https://doi.org/10.1145/3331447.
Bertolino, A., 2007. Software testing research: achievements, challenges, dreams. FoSE 2007 Futur. Softw. Eng. (September), 85–103. https://doi.org/10.1109/FOSE.2007.25.
Chakraborty, S.S., Shah, V., 2011. Towards an approach and framework for test-execution plan derivation. In: 2011 26th IEEE/ACM International Conference on Automated Software Engineering, ASE 2011, Proceedings. IEEE, pp. 488–491. https://doi.org/10.1109/ASE.2011.6100106.
Digital Ocean, "DigitalOcean | The Cloud for Builders." Accessed: May 26, 2023. [Online]. Available: https://www.digitalocean.com/.
ElasTest EU Project, 2017. Fullteaching: A Web Application to Make Teaching Online Easy. Universidad Rey Juan Carlos. Accessed: Aug. 10, 2023. [Online]. Available: https://github.com/elastest/full-teaching.
Esfahani, H., et al., 2016. CloudBuild: microsoft's distributed and caching build service. In: *Proceedings - International Conference on Software Engineering*. ACM, pp. 11–20. https://doi.org/10.1145/2889160.2889222 {ICSE} '16.
Fallahzadeh, E., Bavand, A.H., and Rigby, P.C., "Accelerating continuous integration with parallel batch testing," 2023, https://doi.org/10.1145/3611643.3616255.
Frajtak, K., Cerny, T., 2022. On persistent implications of E2E testing. Lect. Notes Bus. Inf. Process. 455 LNBIP (January), 326–338. https://doi.org/10.1007/978-3-031-08965-7_16.
Gambi, A., Gorla, A., Zeller, A., 2017. O!Snap: cost-efficient testing in the cloud. In: *Proc. - 10th IEEE Int. Conf. Softw. Testing, Verif. Validation, ICST 2017*, pp. 454–459. https://doi.org/10.1109/ICST.2017.51.
García Galán, J., "Automating the support of highly-configurable services," 2015, https://doi.org/10.13140/RG.2.1.3554.9281.
García-Galán, J., García, J.M., Trinidad, P., Fernández, P., 2017. Modelling and analysing highly-configurable services. In: ACM International Conference Proceeding Series, pp. 114–122. https://doi.org/10.1145/3106195.3106211. Sevilla.
Garcia, B., et al., 2018. A proposal to orchestrate test cases. In: *Proceedings - 2018 International Conference on the Quality of Information and Communications Technology, QUATIC 2018*, pp. 38–46. https://doi.org/10.1109/QUATIC.2018.00016.
Gene, D.R., Amdahl, M., 1967. Validity of the single processor approach to achieving large scale computing capabilities. In: *AFIPS Conf. Proc. - 1967 Spring Jt. Comput. Conf. AFIPS 1967*, pp. 483–485. https://doi.org/10.1145/1465482.1465560. Apr.
Google, "Google cloud platform pricing calculator," Google. Accessed: Jun. 08, 2022. [Online]. Available: https://cloud.google.com/products/calculator.
Google, "Google Cloud Computing Services," Google. Accessed: May 26, 2023. [Online]. Available: https://cloud.google.com/.
Greca, R., Miranda, B., Bertolino, A., 2023. State of practical applicability of regression testing research: a live systematic literature review. ACM Comput. Surv. https://doi.org/10.1145/3579851. Dec.
Gyori, A., Shi, A., Hariri, F., Marinov, D., 2015. Reliable testing: detecting state-polluting tests to prevent test dependency. *2015 Int. Symp. Softw. Test. Anal. ISSTA 2015 - Proc.* 223–233. https://doi.org/10.1145/2771783.2771793.
IBM, "IBM Cloud." Accessed: May 26, 2023. [Online]. Available: https://www.ibm.com/ru-ru/cloud.
Inçki, K., Ari, I., Sözer, H., 2012. A survey of software testing in the cloud. In: Proc. 2012 IEEE 6th Int. Conf. Softw. Secur. Reliab. Companion, SERE-C 2012, pp. 18–23. https://doi.org/10.1109/SERE-C.2012.32.
Janakiram MSV, "Lightning Fast Container Provisioning with Microsoft's Azure Container Instances," The New Stack. Accessed: Aug. 09, 2023. [Online]. Available: https://thenewstack.io/lightning-fast-container-provisioning-with-microsofts-azure-container-instances/.
Khajeh-Hosseini, A., Sommerville, I., Bogaerts, J., Teregowda, P., 2011. Decision support tools for cloud migration in the enterprise. In: *Proc. - 2011 IEEE 4th Int. Conf. Cloud Comput. CLOUD 2011*, pp. 541–548. https://doi.org/10.1109/CLOUD.2011.59.
Koskinen, M., Mikkonen, T., Abrahamsson, P., 2019. Containers in Software Development: A Systematic Mapping Study. Springer International Publishing. https://doi.org/10.1007/978-3-030-35333-9_13 vol. 11915 LNCS.
Li, X., Li, Y., Liu, T., Qiu, J., Wang, F., 2009. The method and tool of cost analysis for cloud computing. In: CLOUD 2009 - 2009 IEEE Int. Conf. Cloud Comput, pp. 93–100. https://doi.org/10.1109/CLOUD.2009.84.
Lian, Y., Le, Z., Huiru, X., Yu, S., Wei, Z., Jun, Z., 2009. A framework of testing as a service. In: *Proc. - Int. Conf. Manag. Serv. Sci. MASS 2009*. https://doi.org/10.1109/ICMSS.2009.5302717.
Memon, A., et al., 2017. Taming google-scale continuous testing. In: *Proceedings - 2017 IEEE/ACM 39th International Conference on Software Engineering: Software Engineering in Practice Track, ICSE-SEIP 2017*. IEEE, pp. 233–242. https://doi.org/10.1109/ICSE-SEIP.2017.16. May.
Mezni, H., Abdeljaoued, T., 2018. A cloud services recommendation system based on fuzzy formal concept analysis. Data Knowl. Eng. 116, 100–123. https://doi.org/10.1016/J.DATAK.2018.05.008. Jul.
Microsoft, "Pricing calculator microsoft azure." Accessed: Jun. 08, 2022. [Online]. Available: https://azure.microsoft.com/en-us/pricing/calculator.
Microsoft, "Azure Container Instances." Accessed: Jun. 21, 2023. [Online]. Available: https://azure.microsoft.com/en-us/services/container-instances/.
Najafi, A., Rigby, P.C., Shang, W., 2019. Bisecting commits and modeling commit risk during testing. In: ESEC/FSE2019 - Proc. 2019 27th ACM Jt. Meet. Eur. Softw. Eng. Conf. Symp. Found. Softw. Eng., pp. 279–289. https://doi.org/10.1145/3338906.3338944.
Okraszewski, M., "Cloudorado." Accessed: Jun. 08, 2022. [Online]. Available: https://www.cloudorado.com/.
Parveen, T., Tilley, S., 2010. When to migrate software testing to the cloud?. In: *ICSTW2010 - 3rd Int. Conf. Softw. Testing, Verif. Valid. Work.*, no. Vm, pp. 424–427. https://doi.org/10.1109/ICSTW.2010.77.
Piraghaj, S.F., Dastjerdi, A.V., Calheiros, R.N., Buyya, R., 2017. ContainerCloudSim: an environment for modeling and simulation of containers in cloud data centers. Softw. Pract. Exp. 47 (4), 505–521. https://doi.org/10.1002/SPE.2422. Apr.
Plewnia, C., 2021. An integrated approach for cloud computing service selection and cost estimation. In: ACM International Conference Proceeding Series. Association for Computing Machinery. https://doi.org/10.1145/3492323.3503505. Dec.
Riungu, L.M., Taipale, O., Smolander, K., 2010. Research issues for software testing in the cloud. In: *Proc. - 2nd IEEE Int. Conf. Cloud Comput. Technol. Sci. CloudCom 2010*, pp. 557–564. https://doi.org/10.1109/CloudCom.2010.58.
Riungu-Kalliosaari, L., Taipale, O., Smolander, K., 2012. Testing in the cloud: exploring the practice. IEEE Softw 29 (2), 46–51. https://doi.org/10.1109/MS.2011.132.
Rothermel, G., Harrold, M.J., Von Ronne, J., Hong, C., 2002. Empirical studies of test-suite reduction. Softw. Test. Verif. Reliab. 12 (4), 219–249. https://doi.org/10.1002/stvr.256.
Seybold, D., Domaschka, J., 2017. Is distributed database evaluation cloud-ready? Communications in Computer and Information Science. Springer Verlag, pp. 100–108. https://doi.org/10.1007/978-3-319-67162-8_12.
Truong, H.L., Dustdar, S., 2010. Composable cost estimation and monitoring for computational applications in cloud computing environments. Procedia Comput. Sci. 1 (1), 2175–2184. https://doi.org/10.1016/j.procs.2010.04.243.
Weerasiri, D., Barukh, M.C., Benatallah, B., Sheng, Q.Z., Ranjan, R., 2017. A taxonomy and survey of cloud resource orchestration techniques. ACM Comput. Surv. 50 (2). https://doi.org/10.1145/3054177.
Wohlin, C., Rainer, A., 2022. Is it a case study?—A critical analysis and guidance. J. Syst. Softw. 192, 111395. https://doi.org/10.1016/j.jss.2022.111395.
Wohlin, C. Runeson, P., Höst, M., Ohlsson, M.C., Regnell, B., and Wesslén, A., *Experiment. Software Eng.*, vol. 9783642290. 2012. https://doi.org/10.1007/978-3-642-29044-2.
Wong, W.E., Morgan, J.R., London, S., Mathur, A.P., 1998. Effect of test set minimization on fault detection effectiveness. *Softw. - Pract. Exp.* 28 (4), 347–369. https://doi.org/10.1002/(SICI)1097-024X(19980410)28:4<347::AID−SPE145>3.0.CO;2-L.
Xie, P., Yang, D., 2018. Research on scheduling of software cloud testing. In: 2017 Int. Conf. Comput. Syst. Electron. Control. ICCSEC 2017, pp. 1311–1314. https://doi.org/10.1109/ICCSEC.2017.8446709.
Yoo, S., Harman, M., 2012. Regression testing minimization, selection and prioritization: a survey. In: Software Testing Verification and Reliability, 22. John Wiley and Sons Ltd., pp. 67–120. https://doi.org/10.1002/stv.430. Mar.
Yu, L., Su, Y., Wang, Q., 2009. Scheduling test execution of WBEM applications. In: *Proceedings - Asia-Pacific Software Engineering Conference, APSEC09*, pp. 323–330. https://doi.org/10.1109/APSEC.2009.27. Batu Ferringhi, Penang, Malaysia.
Yu, L., et al., 2010. Testing as a service over cloud. In: *Proceedings - 5th IEEE International Symposium on Service-Oriented System Engineering, SOSE 2010*, pp. 181–188. https://doi.org/10.1109/SOSE.2010.36. Nanjing, China.
Zhang, M., Ranjan, R., Nepal, S., Menzel, M., Haller, A., 2012. A declarative recommender system for cloud infrastructure services selection. Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics) 7714 LNCS, 102–113. https://doi.org/10.1007/978-3-642-35194-5_8.

Cristian Augusto is an Assistant Professor at the Department of Computer Science of the University of Oviedo, Asturias, Spain. Augusto has been teaching courses in software engineering at various levels and he is part of the GIIS Research group. His research interests include Software Verification and Validation and Software Testing concisely, resource optimization during the End-to-End test suite execution and privacy-preserving data publishing techniques.



Jesús Moran received the Ph.D. degree in computing from the University of Oviedo. He is an Assistant Professor with the University of Oviedo. He is a Member of the Software Engineering Research Group (GIIS, giis.uniovi.es). His research interests include software testing, Big Data technologies, and distributed programming.

Antonia Bertolino She is a Research Director of the Italian National Research Council (CNR), at ISTI, Pisa, Italy. Her research covers software validation and testing, on which she worked in several national and European projects. Currently she is Area Chair for Elsevier Journal of Systems and Software, and Associate Editor for Wiley Journal of Software: Evolution and Process. She serves regularly in the PC of top Software Engineering and Testing conferences, such as ESEC-FSE, ASE, ICSE, ISSTA and ICST. She was the General Chair of ACM/IEEE ICSE2015 in Florence (Italy). She has co-authored over 200 papers in international journals and conferences.



Claudio de la Riva is an Associate Professor of the Department of Computer Science, University of Oviedo, Spain. He has received the Ph.D. degree from the Department of Computer Science, University of Oviedo, Spain, in 2004. His research interests include Software Verification and Validation and Software Testing.

Javier Tuya is an Professor of the Department of Computer Science, University of Oviedo, Spain. He has received the Ph.D. degree from the Department of Electrical Engineering, University of Oviedo, Spain, in 1995. He is a member of the ACM, IEEE and the Computer Society. His research interests include Software Quality Assurance, Process Improvement, Verification and Validation and Software Testing.